INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Learning Algorithms for Restricted Neural Networks

by:
Saeed Hadjifaradji

Thesis
submitted to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Philosophy Doctorate (Ph.D.) in Physics

Ottawa-Carleton Institute for Physics
University of Ottawa
Ottawa, Canada

September 1999

Abstract

The problem of supervised learning can be phrased in terms of finding a good approximation to some unknown target function f, based on observing f's values on some examples drawn randomly according to some underlying distribution. Unfortunately, finding an algorithm that efficiently learns simple classes of neural networks under any distribution has proven to be very hard. In this thesis, we propose learning algorithms that *provably* and *efficiently* learn simple classes of neural networks under simple distributions within the rigorous framework of computational learning theory. We therefore propose some restrictions on both the neural network class and the underlying distribution that generates the examples because, otherwise, the learning problem becomes intractable.

In the first part of the thesis, we present an algorithm that provably learns the class of stochastic perceptrons with arbitrary monotonic activation function and binary weights when the probability distribution that generates the input examples is member of a class that we call *k-blocking distributions*.

The problem studied in Second part of the thesis is much more involved. Whereas in the first part, the target function consists only of a single perceptron, in the second part, it consist of a nonoverlapping network of perceptrons. We provide an algorithm that provably finds both the binary-valued weights and the network connectivity of the target function when the distribution is such that each variable is statistically independent of all the others.

Acknowledgments

I would like to express my gratitude to my research advisor, Dr. Mario Marchand, for his excellent guidance and supervision provided throughout all the steps of this research. Without his continuing help, this thesis could not have been written. I would like also to thank him for his financial support through his NSERC research grant.

I would like also to thank the faculty members at the Department of Physics. In particular, I thank Dr. Béla Joos, the Department Chairman, and all the staff and graduate students for their friendship. In particular, I thank Grant Nixon and his family on whom I could always count.

Also, I thank Department of Computer Science for its access to the grad lab and facilities.

As well, I would like to thank the Minister of Culture and Higher Education of Iran that financially supported me for four years.

At last but not least, I would like to thank my family who supported me at all time.

Contents

List of Figures

Chapter 1

Introduction

Research in artificial neural networks now constitutes a significant portion of the current research in artificial intelligence. Successes in applications as diverse as financial forecasting, optical character recognition, and robotics are now numerous. In most of these applications a finite *training set* $\{< \mathbf{x}_i, y_i >\}_{i=1}^{m}$ of m examples is first gathered. Here $\mathbf{x}_i$ refers to a n-dimensional input vector of values for which the desired scalar output is y_i. In other words, y_i is the output value of some unknown function f evaluated on input example $\mathbf{x}_i$ *i.e.* $y_i = f(\mathbf{x}_i)$ for $i = 1 \cdots m$. A *learning algorithm* then reads this training set of examples and eventually produces a neural network function $h(\mathbf{x})$. For now it suffices to say that this hypothesis function $h(\mathbf{x})$ is represented as a network of simple computing elements that are called neurons. Given any input vector $\mathbf{x}$, the neural network computes an output value $h(\mathbf{x})$. The goal of the learning algorithm is to produce a hypothesis neural network function $h(\mathbf{x})$ which will agree most of the time with $f(\mathbf{x})$ on future examples $\mathbf{x}$ not present in the training set. If this is the case we say that h *generalizes* well on new examples. Designing learning algorithms that achieve good generalization is a major objective of both this thesis and all of neural network research.

The methods used to evaluate the generalization performance of a neural network learning algorithm are mainly empirical. Typically, a set of examples is split into two disjoint subsets: the *training set* used by the learning algorithm and the *testing set* used to test the accuracy of the hypothesis h that was returned by the learning algorithm ran on the training set. To obtain a better estimate of the algorithm's performance, the *k-fold cross-validation* method (see [27] for example) is usually performed. For this, the entire set of data is first split into k disjoint subsets. The learning algorithm is trained on the union of $k-1$ subsets and the returned hypothesis h is tested on the remaining subset. The generalization scores are then averaged over the k different testing sets to obtain an estimation of the algorithm's generalization ability. In this way, the application developer can test different learning algorithms and pick the one which performs best on his data. However, satisfactory this might be for the application developer, such empirical evaluations provide no information on the performance of a learning algorithm outside the scope of the application: we simply cannot predict how the algorithm will perform outside the domain in which it was tested. *Hence, what we need from a scientific point of view is a precise characterization of the conditions under which a given learning algorithm will give a certain level of performance.*

Judging from the overwhelming number of different neural network learning algorithms that currently exist compared to those very few for which we can prove something about them, we can say that it is quite easy to design a learning algorithm which is based on "sound principles" that "make sense in practice" but it is much more difficult to obtain a quantitative characterization of the conditions under which this algorithm will work satisfactorily. Since this thesis is concerned only about neural network learning algorithms for which we can provide a precise characterization of the algorithm's performance, we will restrict the discussion to these. But before we can

state the relevant results on to which this thesis is built upon, we need to introduce some basic notions of learning theory.

A *learning algorithm* is an algorithm that takes as input a set of m training examples $\{< \mathbf{x}_i, y_i >\}_{i=1}^{m}$ and returns a function $h(\mathbf{x})$ called the *hypothesis function*. Each $\mathbf{x}_i$ is a n-dimensional vector whose components are restricted to be Boolean valued throughout this thesis. Each y_i, known as the *class label* of input example $\mathbf{x}_i$, is also restricted to be Boolean valued throughout this thesis. Each input example $\mathbf{x}_i$ is generated independently according to some unknown distribution D and then classified according to some unknown function f that we call the target function. Hence, the learner is provided only with the values of f on m points. To simplify the discussion, let us assume for now that the function f to learn and the hypothesis h returned are always deterministic functions, *i.e.* their output value on any input $\mathbf{x}$ is uniquely determined with probability one. In these circumstances, the error $\mathrm{err}(h, f)$ that h makes with f is defined to be the probability that h disagrees with f on a new example $\mathbf{x}$ drawn randomly according to the same (but unknown) distribution D that generated the training set

$$\mathrm{err}(h, f) \stackrel{\mathrm{def}}{=} \mathrm{Pr}_{\mathbf{x} \in D} \{h(\mathbf{x}) \neq f(\mathbf{x})\} \tag{1.1}$$

Given a set of m examples, the goal of the learner is to return a hypothesis for which $\mathrm{err}(h, f)$ is as small as possible. However, h depends on the training set that was used by the learner: if we change the training set, h will generally change (for the same learning algorithm). Hence, a reasonable goal for the learner will be to return *with high probability* (over the different training sets) a hypothesis that makes small error. Moreover, we should not expect that a learning algorithm will achieve that goal for any target function f and any distribution D but rather only for a restricted class F of functions and a restricted family $\mathcal{D}$ of distributions. By identifying F and $\mathcal{D}$, we

are in fact providing precise conditions under which a learning algorithm will perform satisfactorily. These considerations have led us to adopt the Probably Approximately Correct (PAC) learning criteria that was proposed by Valiant [30].

Definition 1.1 *A learning algorithm A is said to PAC learn a class F of functions under a family $\mathcal{D}$ of distribution iff for any $f \in F$, any $D \in \mathcal{D}$, any $0 < \epsilon < 1$, and any $0 < \delta < 1$, we have that A returns h such that* $\mathrm{err}(h, f) < \epsilon$ *with probability at least $1 - \delta$ over the different training sets of size m. The smallest $m(\epsilon, \delta)$ satisfying this property is called the sample complexity of algorithm A.*

Any learning algorithm must choose the returned hypothesis h among a set H of functions that it is capable of implementing. Vapnik [32, 33, 34] has made an enormous contribution to learning theory by identifying a key property of H, called the *VC-dimension* of H, noted by $VC(H)$, which affects the sample complexity of learning algorithms. The $VC(H)$ is the maximum number d of points $\{\mathbf{x}_i\}_{i=1}^{d}$ for which we are capable of producing all the 2^d different Boolean vectors $(h(\mathbf{x}_1), h(\mathbf{x}_2), \cdots, h(\mathbf{x}_d))$ by using only the functions h in the hypothesis space H. Indeed, from Vapnik's work it is possible to show [1] that *any* learning algorithm that can always find an $h \in H \supseteq F$ that makes zero empirical error on the training set (*i.e.* $h(\mathbf{x}_i) = f(\mathbf{x}_i)$ for $i = 1, \cdots, m$) will need at most that number of examples

$$m(\epsilon, \delta) = \frac{8}{\epsilon}\left(\ln\left(\frac{4}{\delta}\right) + VC(H)\ln\left(\frac{48}{\epsilon}\right)\right)$$

to PAC learn the class F with the hypothesis class H.

A learning algorithm that can always return an h that makes zero empirical error with a training set is said to be *consistent*. Hence, if $F \subseteq H$ and if the $VC(H)$ is finite, then *any* consistent learning algorithm that uses H is a PAC learning algorithm for F and the number of examples needed just grows linearly with $VC(H)$.

A neural network learning algorithm usually restricts itself to a class H of functions that are representable as a restricted set of neural networks. The most restricted set is the class of functions representable as a single neuron. Here by neurons, or perceptrons, we mean linear threshold devices. Hence, a perceptron h on $\mathbf{x}$ with weight vector $\mathbf{w}$ and threshold θ outputs $h(\mathbf{x}) = 1$ if $\sum_{j=1}^{n} w_j x_j > \theta$, otherwise $h(\mathbf{x}) = 0$. Figure 1.1 illustrates the computation performed by a single perceptron. The VC-dimension of a single perceptron with n real valued weights and a threshold is $n + 1$ (see [1] for a simple proof). Hence, the number of examples sufficient to PAC learn a perceptron is linear in n.

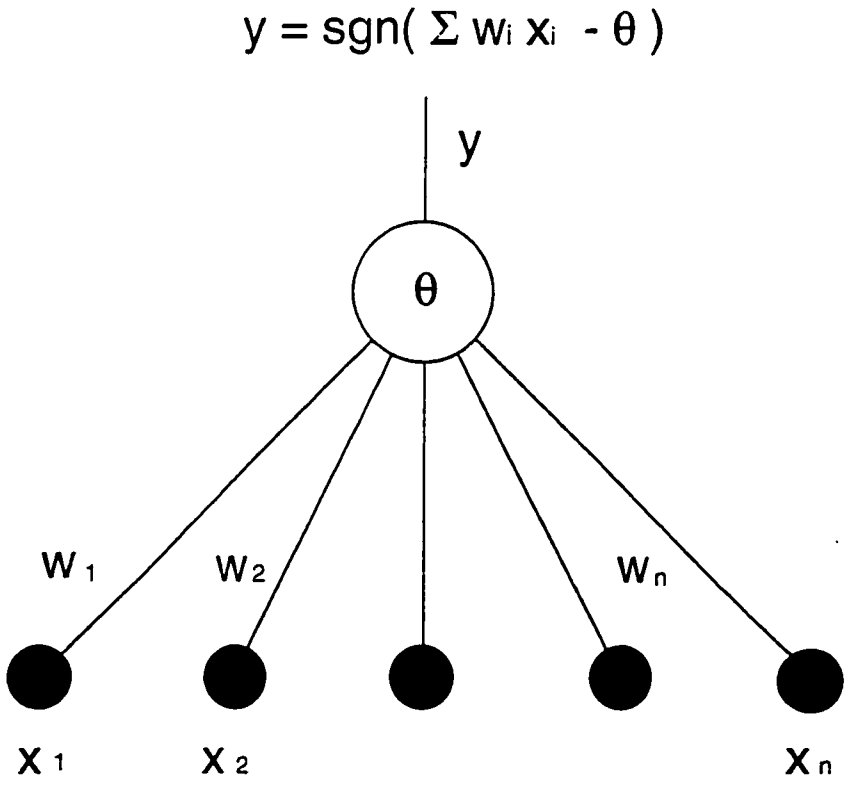

Figure 1.1: A single perceptron. The function $\mathrm{sgn}(x)$ is 1 if $x > 0$ and 0 otherwise.

A neural network is a network of perceptrons. A *feedforward* neural network is a network of perceptrons with no feedback connections. Figure 1.2 illustrates the

computation performed by a multi-layered feedforward neural network. Such a network consists of input units (shown in black), hidden units, and a single output unit. Every unit is a perceptron and performs the computation stated earlier (i.e. as in figure 1.1). When an input example is presented to the input units. The ouput values of the (hidden) perceptrons in the first layer is first computed. Then the ouput values of the perceptrons in the second hidden layer is computed; and so on, until the output unit can compute its output value. This last value is the output value of the whole network for the given input example.

Baum and Haussler [4] have shown that the VC-dimension for such networks with W variable weights and thresholds is less than $6W \log W$. Hence, the sufficient number of examples needed for any consistent feedforward neural network learning algorithm is obtained by substituting $6W \log W$ in place of $VC(H)$ in the above formula for $m(\epsilon, \delta)$. The problem we are facing here is that we *need* a consistent algorithm: one that can always find the values of all the weights (and thresholds) such that the neural network is consistent with the training set. The standard algorithm used by the practitioners to train feedforward neural networks is a gradient descent algorithm known as back-propagation [35, 26]. However, with gradient descent, one might get trapped in some local minima where consistency does not occur. In addition, it is possible to have exponentially many local minima even for a single perceptron [3]. Hence, gradient descent (and variants) might take a prohibitive amount of training time even for the simplest feedforward neural nets.

Hence, the above discussion shows that we want more for an algorithm than just being able to PAC learn, we want it to be *efficient*, *i.e.* we want its *training time* to increase no more than a polynomial in terms of all the relevant parameters that characterize the hypothesis class H used by the algorithm (like $VC(H)$), the precision

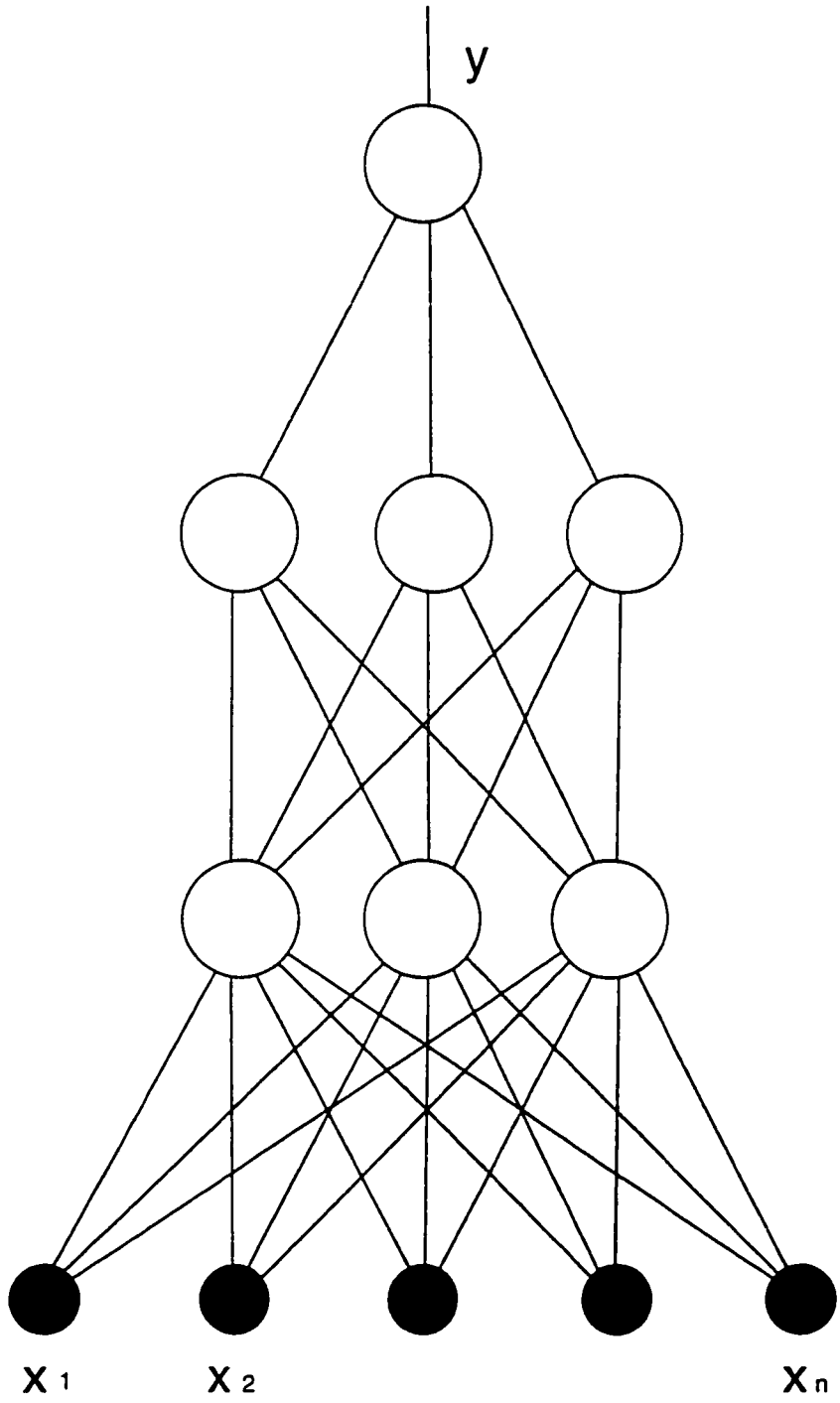

Figure 1.2: A multi-layered feedforward neural network.

ϵ that we require for h and the confidence parameter δ. If this is the case we will say that algorithm A is an *efficient* PAC learning algorithm for class F under H.

Training a single perceptron with real weights is a sub problem of Linear Programming. Hence, we can use any polynomial time linear programming algorithm to efficiently PAC learn the class F of real weight perceptrons with the same hypothesis class $H = F$. But this is basically using a "jack-hammer to push a nail". In fact, there exists a very simple, on-line, adaptive rule known as the *Perceptron Learning Rule*, originally proposed by Rosenblatt [25]. which takes at most $(R/\gamma)^2$ additive updates in order to find a weight vector and threshold that is consistent with any given training set of m examples which are enclosed in a ball of radius R and for which there exists a consistent separating hyperplane with margin γ (see Block [5]). Hence, for this simple learning rule the time is not polynomial in the number n of weights but deteriorates polynomially in terms of R/γ.

Suppose that the function to learn is a perceptron with "binary" weights, *i.e.* the weights are restricted to be in $\{-1, 0, +1\}$. From the above discussion, we know that we can certainly efficiently PAC learn this class with the hypothesis class consisting of real weight perceptrons. But can we learn binary perceptrons with binary perceptrons? To answer this, we note that the consistency problem of determining a set of binary weights that will linearly separate a training set of examples is an Integer Programming problem. Hence, unless $P = NP$, we cannot efficiently PAC learn binary perceptrons with binary perceptrons under an arbitrary distribution D. But the hardness of the problem changes substantially if the distribution D that generates the examples is restricted to some well defined class. Indeed, Golea and Marchand [7] have shown that a simple adaptive rule, known as the *clipped-Hebb rule*, is able to efficiently PAC learn binary perceptrons under the uniform distribution. Moreover, they have shown that a

simple modification of this rule can handle the case of binary perceptrons when D is a member of the family of *product distributions*: *i.e.* distributions for which each input variable is statistically independent of all the others.

In the next chapter, we show how a modification of the clipped-Hebb rule can learn stochastic perceptrons with binary weights under a larger family of distributions that we call *k-blocking distributions*. By a stochastic perceptron, we mean a perceptron whose Boolean output value is determined stochastically: given an input vector $\mathbf{x}$, the output is equal to one with the probability given by $f(\sum_{i=1}^{n} w_i x_i)$ where $f(.)$ is an arbitrary monotonic function usually called the *activation function*. However, it is clear that we cannot find a hypothesis whose accuracy can be arbitrarily close to such a stochastic function. In this case, we can only hope to find a good model of probability of its stochastic behavior. Hence, as we will discuss in the next chapter, we need to extend the above mentioned PAC learning criteria to the one proposed by Kearns and Schapire [20]. We have already published elsewhere [22] the results of this chapter.

The third chapter of the thesis is devoted to the harder problem of learning networks of perceptrons. In that respect, Blum and Rivest [6] have shown that the consistency problem of training an AND (or an OR) of just two perceptrons is NP-complete. Although this result does not rule out that such simple networks might be learnable by networks containing more perceptrons. to our knowledge, no efficient algorithm exists for this problem. However, we can simplify the network class by removing the overlap between the receptive fields of the perceptrons. By using the techniques of Kearns *et al.* [17], Golea and Marchand [8] have shown that the consistency problem of learning an AND/OR of two nonoverlapping perceptrons is as hard as that of an AND/OR of overlapping perceptrons. More generally, this shows that, under arbitrary input distributions, the problem of learning nonoverlapping networks is as hard as learning

completely overlapping networks. But again, restricting the input distribution changes the difficulty of the learning problem. Indeed, Golea, Marchand and Hancock [10] provided algorithms to efficiently PAC learn, under the uniform distribution, AND/OR of nonoverlapping perceptrons and generalized decision lists of nonoverlapping perceptrons. In the remaining chapters of this thesis, we give a polynomial time algorithm that PAC learns the general class of nonoverlapping perceptrons under the family of product distributions. This is the general class of loop free neural networks where each node has only one outgoing weight. In analogy with the class of μ-formulas originally studied by Valiant [30], these nonoverlapping neural networks constitute the class of μ-perceptron networks. An algorithm for learning the restricted case of constant depth μ-perceptron networks with no weakly relevant variables has already been published by us elsewhere [23]. In this thesis, we presents an algorithm that can handle all these simpler cases [23, 10] in addition to the more complex case of arbitrary depth μ-perceptron nets with weakly relevant variables.

Chapter 2

Learning Stochastic Perceptrons Under K-Blocking Distributions

In this chapter, we first introduce a new family of distributions that we call *k-blocking distributions*. Such distributions represent an important step beyond product distributions since the 2k-blocking family contains all the Markov distributions of order k. We then present a statistical method that PAC learns the class of stochastic perceptrons with arbitrary monotonic activation function and binary weights when the probability distribution that generates the input examples is a k-blocking distribution.

2.1 Introduction

The PAC learning criterion [30] has recently been extended [13] to analyze the learnability of *probabilistic* concepts [20, 28, 31]. Such concepts, which are stochastic rules that give the probability that input example $\mathbf{x}$ is classified as being positive [20, 28, 31], are natural probabilistic extensions of the deterministic concepts originally studied by Valiant [30].

Motivated by the stochastic nature of many "real-world" learning problems and by the indisputable fact that biological neurons are probabilistic devices, some preliminary studies about the PAC learnability of simple probabilistic neural concepts have been reported [7, 9] recently. However, the probabilistic behaviors considered in these studies are quite specific and clearly need to be extended. Indeed, only classification noise superimposed on a deterministic signum function was considered in [7]. The probabilistic network analyzed in [9], consists of a linear superposition of signum functions and is thus solvable as a (simple) case of linear regression. What is clearly needed is the extension to the non-linear cases of sigmoïds and radial basis functions. Another criticism about [7, 9] is the fact that their learnability results was established only for distributions where each input variable is statistically independent from all the others (product distributions). In fact, very few positive learning results for non-trivial p-concepts classes are known to hold for larger classes of distributions. Therefore, in an effort to find algorithms that will work in practice, we introduce in this research a new family of distributions that we call *k-blocking* distributions. As we will argue, this family has the dual advantage of avoiding malicious and unnatural distributions that are prone to render simple concept classes unlearnable [21] and, of being likely to contain several distributions found in practice.

Our main contribution is to present a simple statistical method that PAC learns (in polynomial time) the class of stochastic perceptrons with monotonic (but otherwise arbitrary) activation functions and weights $w_i \in \{-1, 0, +1\}$ when the input examples are generated according to any distribution member of the k-blocking family.

In the next section, we will introduce some preliminary definitions and present the extension [20] of the PAC learning criterion for learning stochastic perceptrons. In the following sections, we will introduce the notion of k-blocking distributions and then

provide the learning algorithm for stochastic perceptrons.

2.2 Preliminary definitions and the criterion for learning

The input space, $\mathcal{I}^n$, is the Boolean domain $\{-1,+1\}^n$. The set of all input variables is denoted by X. Each input example $\mathbf{x}$ is generated according to some unknown distribution D on $\mathcal{I}^n$. As usual, we will use $p_D(\mathbf{x})$, or simply $p(\mathbf{x})$, to denote the probability of observing the vector value $\mathbf{x}$ under distribution D. If U and V are two disjoint subsets of X, $\mathbf{x}_U$ and $\mathbf{x}_V$ will denote the restriction (or projection) of $\mathbf{x}$ over the variables of U and V respectively and $p_D(\mathbf{x}_U|\mathbf{x}_V)$ will denote the probability, under distribution D, of observing the vector value $\mathbf{x}_U$ (for the variables in U) given that the variables in V are set to the vector value $\mathbf{x}_V$.

Following Kearns and Schapire [20], a *probabilistic concept* (p-concept) is a map $c : \mathcal{I}^n \to [0,1]$ for which $c(\mathbf{x})$ represents the probability that example $\mathbf{x}$ is classified as positive. More precisely, upon presentation of input $\mathbf{x}$, an output of $\sigma = 1$ is generated (by an unknown target p-concept) with probability $c(\mathbf{x})$ and an output of $\sigma = 0$ is generated with probability $1 - c(\mathbf{x})$.

A *stochastic perceptron* is a p-concept parameterized by a vector of n weights w_i and a *activation function* $f(\cdot)$ such that, the probability that input example $\mathbf{x}$ is classified as positive is given by

$$\Pr(\sigma = 1 \,|\mathbf{x}) \;=\; f\left(\sum_{i=1}^{n} w_i x_i\right) \quad . \tag{2.1}$$

We restrict ourselves to the case where $f(\cdot)$ is *monotonic i.e.* either nondecreasing or nonincreasing. Moreover, the learner can assume without loss of generality that the

target stochastic perceptron has a nondecreasing $f(\cdot)$. Indeed, any stochastic perceptron with a nonincreasing $f(\cdot)$ and weight vector $\mathbf{w}$ can always be represented with a nondecreasing activation function and weight vector $-\mathbf{w}$. Hence, we allow any sigmoïd-type of activation function (with arbitrary threshold). Also, since our instance space $\mathcal{I}^n$ is on an n-sphere, eq. 2.1 also includes any nonincreasing radial basis function of the type $\phi(z^2)$ where $z = |\mathbf{x} - \mathbf{w}|$ and $\mathbf{w}$ is interpreted as "center" of ϕ. The only significant restriction is on the weights where we allow only for $w_i \in \{-1, 0, +1\}$.

As usual, the goal of the learner is to return a *hypothesis* h which is a good approximation of the target p-concept c. But, in contrast with *decision rule learning* which attempts to "filter out" the noisy behavior by returning a deterministic hypothesis, the learner will attempt the harder (and more useful) task of modeling the target p-concept by returning a p-concept hypothesis. As a measure of error between the target and the hypothesis p-concepts we adopt the *variation distance* $d_v(\cdot, \cdot)$ defined as:

$$err(h, c) \;=\; d_v(h, c) \stackrel{\text{def}}{=} \sum_{\mathbf{x}} p_D(\mathbf{x}) \, |h(\mathbf{x}) - c(\mathbf{x})| \tag{2.2}$$

Where the summation is over all the 2^n possible values of $\mathbf{x}$. Hence, the same D is used for both training and testing. The following formulation of the PAC criterion [30, 13, 20] will be sufficient for our purpose.

Definition 2.1 *Algorithm A is said to PAC learn the class C of p-concepts by using the hypothesis class H (of p-concepts) under a family $\mathcal{D}$ of distributions on instance space $\mathcal{I}^n$,* iff *for any $c \in C$, any $D \in \mathcal{D}$, any $0 < \epsilon, \delta < 1$, algorithm A returns in a time polynomial in $(1/\epsilon, 1/\delta, n)$, a hypothesis $h \in H$ such that with probability at least $1 - \delta$, $err(h, c) < \epsilon$.*

2.3 A reasonable distribution class: the k-blocking family

To learn the class of stochastic perceptrons, the algorithm will try to discover each weight w_i that connects to input variable x_i by estimating how the probability of observing a positive output ($\sigma = 1$) is affected by "hard-wiring" variable x_i to some fixed value. This should clearly give some information about w_i when x_i is statistically independent from all the other variables as was the case for [7, 28]. However, if the input variables are correlated, then the process of fixing variable x_i will carry over neighboring variables which in turn will affect other variables until all the variables are perturbed (even in the simplest case of a first order Markov chain). The information about w_i will then be smeared by all the other weights. Therefore, to obtain information only on w_i, we need to break this "chain reaction" by fixing some other variables. The notion of *blocking sets* serves this purpose.

Loosely speaking, a set of variables is said to be a blocking set[1] for variable x_i if the distribution on all the remaining variables is unaffected by the setting of x_i whenever all the variables of the blocking set are set to a fixed value. More precisely, we have:

Definition 2.2 *Let B be a subset of X and let $U = X - (B \cup \{x_i\})$. Let $\mathbf{x}_B$ and $\mathbf{x}_U$ be the restriction of $\mathbf{x}$ on B and U respectively and let $\mathbf{b}$ be an assignment for $\mathbf{x}_B$. Then B is said to be a blocking set for variable x_i (with respect to D),* iff*:*

$$p_D(\mathbf{x}_U|\mathbf{x}_B = \mathbf{b}, x_i = +1) = p_D(\mathbf{x}_U|\mathbf{x}_B = \mathbf{b}, x_i = -1) \quad \text{for all } \mathbf{b} \text{ and } \mathbf{x}_U \tag{2.3}$$

In addition, if B is not anymore a blocking set when we remove anyone of its variables,

[1]The wording "blocking set" was also used by Hancock & Mansour (*Proc. of COLT'91*, 179–183, Morgan Kaufmann Publ.) to denote a property of the target concept. In contrast, our definition of blocking set denotes a property of the input distribution only.

we then say that B is a minimal *blocking set for variable x_i.*

We thus adopt the following definition for the k-blocking family.

Definition 2.3 *Distribution D on $\mathcal{I}^n$ is said to be k-blocking* iff $|B_i| \leq k$ for $i = 1, 2 \cdots n$ *when each B_i is a minimal blocking set for variable x_i.*

The k-blocking family is quite a large class of distributions. In fact, we have the following property:

Property 2.1 *All Markov distributions of kth order are members of the 2k-blocking family.*

Proof: By kth order Markov distributions, we mean distributions which can be exactly written as a Chow(k) expansion (see [14]) for some permutation of the variables. We prove it here (by using standard techniques [2]) for first order Markov distributions, the generalization for $k > 1$ is straightforward. Recall that for Markov chain distributions we have: $p(x_j|x_{j-r}, \cdots x_1) = p(x_j|x_{j-r})$ for $1 \leq j \leq n$ and $1 \leq r < j$. Hence:

$$\begin{aligned} & p(x_1 \cdots x_{j-2}, x_{j+2} \cdots x_n | x_{j-1}, x_j, x_{j+1}) \\ & \quad = p(x_1)p(x_2|x_1) \cdots p(x_j|x_{j-1})p(x_{j+1}|x_j) \cdots p(x_n|x_{n-1})/p(x_{j-1}, x_j, x_{j+1}) \\ & \quad = p(x_1)p(x_2|x_1) \cdots p(x_{j-1}|x_{j-2})p(x_{j+2}|x_{j+1}) \cdots p(x_n|x_{n-1})/p(x_{j-1}) \\ & \quad = p(x_1 \cdots x_{j-2}, x_{j+2} \cdots x_n | x_{j-1}, \overline{x_j}, x_{j+1}) \end{aligned} \tag{2.4}$$

where $\overline{x_j}$ denotes the negation of x_j. Thus, we see that Markov chain distributions are a special case of 2-blocking distributions: the blocking set of each variable consisting only of the two first-neighbor variables. □

The proposed algorithm for learning stochastic perceptrons needs to be provided with a blocking set (of at most k variables) for each input variable. Hoeffgen [14] has

recently proven that Chow(1) and Chow($k > 1$) expansions are efficiently learnable; the latter under some restricted conditions. We can thus use these algorithms to discover the blocking sets for such distributions. However, the efficient learnability of unrestricted Chow($k > 1$) expansions and larger classes of distributions, such as the k-blocking family, is still unknown. In fact, from the hardness results of Hoeffgen [14], we can see that it is definitely very hard (perhaps NP-complete) to find the blocking sets if the learner has no information available other than the fact that the distribution is k-blocking. On the other hand, we can argue that the "natural" ordering of the variables present in many "real-world" situations is such that the blocking set of any given variable is among the neighboring variables. In vision for example, we expect that the setting of a pixel will directly affect only those located in it's neighborhood; the other pixels being affected only through this neighborhood. In such cases, the neighborhood of a variable "naturally" provides its blocking set.

2.4 Learning stochastic perceptrons

We first establish (the intuitive fact) that, without making much error, we can always consider that the target p-concept is defined only over the variables which are not almost always set to the same value.

Lemma 2.1 *Let U be a set of u variables x_i for which $\Pr(x_i = a_i) > 1 - \alpha$ for some $a_i \in \{0, 1\}$. Let c be a p-concept and let c' be the same p-concept as c except that the reading of each variable $x_i \in U$ is replaced by the reading of the constant value a_i. Then $err(c', c) < u \cdot \alpha$.*

Proof: Let $\mathbf{a}$ be the vector obtained from the concatenation of all a_is and let $\mathbf{x}_U$ be the vector obtained from $\mathbf{x}$ by keeping only the components x_i which are in U. Then:

$$\begin{aligned} err(c',c) &= err(c',c|\mathbf{x}_U = \mathbf{a})\Pr(\mathbf{x}_U = \mathbf{a}) \\ &\quad + err(c',c|\mathbf{x}_U \neq \mathbf{a})\Pr(\mathbf{x}_U \neq \mathbf{a}) \\ &\leq \Pr(\mathbf{x}_U \neq \mathbf{a}) \\ &\leq \sum_{x_i \in U} \Pr(x_i \neq a_i) \\ &\leq u \cdot \alpha. \end{aligned} \tag{2.5}$$

□.

For a given set of blocking sets $\{B_i\}_{i=1}^n$, the algorithm will try to discover each weight w_i by estimating the *blocked influence* of x_i defined as:

$$\begin{aligned} \mathrm{Binf}(x_i|\mathbf{b}_i) &\stackrel{\text{def}}{=} \Pr(\sigma = 1|\mathbf{x}_{B_i} = \mathbf{b}_i, x_i = +1) \\ &\quad - \Pr(\sigma = 1|\mathbf{x}_{B_i} = \mathbf{b}_i, x_i = -1) \end{aligned} \tag{2.6}$$

where $\mathbf{x}_{B_i}$ denotes the restriction of $\mathbf{x}$ on the blocking set B_i for variable x_i and $\mathbf{b}_i$ is an assignment for $\mathbf{x}_{B_i}$. The following lemma ensures the learner that $\mathrm{Binf}(x_i|\mathbf{b}_i)$ contains enough information about w_i.

Lemma 2.2 *Let the target p-concept be a stochastic perceptron on $\mathcal{I}$ having a non-decreasing activation function and weights taken from $\{-1, 0, +1\}$. Then, for any assignment $\mathbf{b}_i$ for the variables in the blocking set B_i of variable x_i, we have:*

$$\mathrm{Binf}(x_i|\mathbf{b}_i) \begin{cases} \geq 0 & \text{if } w_i = +1 \\ = 0 & \text{if } w_i = 0 \\ \leq 0 & \text{if } w_i = -1 \end{cases} \tag{2.7}$$

Proof: Let $W = X - (B_i \cup \{x_i\})$, $s = \sum_{x_j \in W} w_j x_j$ and $\zeta(\mathbf{b}_i) = \sum_{k \in B_i} w_k b_k$. Let $p(\mathbf{x}_W|\mathbf{b}_i)$ denote the probability of observing the restriction $\mathbf{x}_W$ (under D) given $\mathbf{b}_i$.

Then:

$$\text{Binf}(x_i|\mathbf{b}_i) = \sum_{\mathbf{x}_W} p(\mathbf{x}_W|\mathbf{b}_i)\left[f(s+\zeta(\mathbf{b}_i)+w_i) - f(s+\zeta(\mathbf{b}_i)-w_i)\right]; \qquad (2.8)$$

from which we find the desired result for any nondecreasing $f(\cdot)$.

□.

In principle, lemma 2.2 enables the learner to discover w_i from $\text{Binf}(x_i|\mathbf{b}_i)$. The learner, however, has only access to its *empirical estimate* $\hat{\text{Binf}}(x_i|\mathbf{b}_i)$ from a finite sample. Hence, we will use Hoeffding's inequality [15] to find the number of examples needed for a probability p to be close to its empirical estimate $\hat{p}$ with high probability.

Lemma 2.3 ([15]) *Let $Y_1, \ldots, Y_m$ be a sequence of m independent Bernoulli trials, each succeeding with probability p. Let $\hat{p} = \sum_{i=1}^{m} Y_i/m$. Then:*

$$\Pr\left(|\hat{p} - p| > \epsilon\right) \;\leq\; 2\exp\left(-2m\epsilon^2\right) \qquad (2.9)$$

Hence, by writing $\text{Binf}(x_i|\mathbf{b}_i)$ in terms of probabilities that can be estimated from *all* the training examples, we find from lemma 2.3 that the number $m_0(\epsilon, \delta, n)$ of examples needed to have $|\hat{\text{Binf}}(x_i|\mathbf{b}_i) - \text{Binf}(x_i|\mathbf{b}_i)| < \epsilon$ with probability at least $1-\delta$ is given by:

$$m_0(\epsilon, \delta, n) \;\geq\; \frac{1}{2}\left(\frac{8}{\kappa\epsilon}\right)^2 \ln\left(\frac{8}{\delta}\right) \qquad (2.10)$$

where $\kappa = \alpha^{k+1}$ is the lowest permissible value for $p_D(\mathbf{b}_i, x_i)$ (see appendix A for details). So, if the minimal nonzero value for $|\text{Binf}(x_i|\mathbf{b}_i)|$ is β, then the number of examples needed to find, with confidence at least $1-\delta$, the exact value for w_i among $\{-1, 0, +1\}$ is such that we need to have: $\Pr(|\hat{\text{Binf}}(x_i|\mathbf{b}_i) - \text{Binf}(x_i|\mathbf{b}_i)| < \beta/2) > 1-\delta$. Thus, whenever β is of $\Omega(e^{-n})$, we will need of $O(e^{2n})$ examples to find (with probability $> 1-\delta$) the value for w_i. So, in order to be able to PAC learn from a polynomial

sample, we must arrange ourselves so that we do not need to worry about such low values for $|\text{Binf}(x_i|\mathbf{b}_i)|$. We, therefore, consider the *maximum blocked influence* defined as:

$$\text{Binf}(x_i) \stackrel{\text{def}}{=} \text{Binf}(x_i|\mathbf{b}_i^*) \tag{2.11}$$

where $\mathbf{b}_i^*$ is the vector value for which $|\text{Binf}(x_i|\mathbf{b}_i)|$ is the largest. We now show that the learner can ignore all variables x_i for which $|\text{Binf}(x_i)|$ is too small (without making much error).

Lemma 2.4 *Let X be the set of all the input variables. Let c be a stochastic perceptron with nondecreasing activation function $f(\sum_j w_j x_j)$ and weights taken from $\{-1, 0, +1\}$. Let $U \subset X$ and let c_U be the same stochastic perceptron as c except that $w_i = 0$ for all $x_i \in U$ and its activation function is changed to $f(\sum_{j \in (X-U)} w_j x_j + \theta)$. Then, there always exists a value for θ such that:*

$$err(c_U, c) \leq \sum_{i \in U} |\text{Binf}(x_i)| \tag{2.12}$$

Proof: Let n be the total number of variables. Let $|U|$ denote the total number of variables in U. Let c_U be the same stochastic perceptron as c except that $w_i = 0$ for all $x_i \in U$ and its activation function is changed to $f(\sum_{j=|U|+1}^{n} w_j x_j + \sum_{j=1}^{|U|} w_j a_j)$ where $a_j \in \{-1, +1\}$.

Now to prove the lemma, we will show that there always exists a setting of values for each $a_j \in \{-1, +1\}$ such that it satisfies inequality 2.12. Therefore, the lemma will be established if we define: $\theta \stackrel{\text{def}}{=} \sum_{j=1}^{|U|} w_j a_j$ for such a setting.

By the definition of the error, we can write:

$$err(c, c_U) = \sum_{\mathbf{x}} \Pr(\mathbf{x}) \left| f(\sum_{j=1}^{n} w_j x_j) - f(\sum_{j=|U|+1}^{n} w_j x_j + \sum_{j=1}^{|U|} w_j a_j) \right| \tag{2.13}$$

where the sum is over all the possible value of $\mathbf{x}$. Now, let's add and subtract terms like $f(\sum_{j=i}^{n} w_j x_j + \sum_{j=1}^{i-1} w_j a_j)$ for $i = 2$ to $i = |U| - 1$. Therefore, we will have:

$$\begin{aligned}
err(c, c_U) &= \sum_{\mathbf{x}} \Pr(\mathbf{x}) \Big| f(\sum_{j=1}^{n} w_j x_j) - f(\sum_{j=2}^{n} w_j x_j + w_1 a_1) \\
&\quad \cdots \\
&\quad + f(\sum_{j=|U|}^{n} w_j x_j + \sum_{j=1}^{|U|-1} w_j a_j) - f(\sum_{j=|U|+1}^{n} w_j x_j + \sum_{j=1}^{|U|} w_j a_j) \Big| \\
&\leq \sum_{\mathbf{x}} \Pr(\mathbf{x}) \Big\{ \Big| f(\sum_{j=1}^{n} w_j x_j) - f(\sum_{j=2}^{n} w_j x_j + w_1 a_1) \Big| \\
&\quad \cdots \\
&\quad \Big| + f(\sum_{j=|U|}^{n} w_j x_j + \sum_{j=1}^{|U|-1} w_j a_j) - f(\sum_{j=|U|+1}^{n} w_j x_j + \sum_{j=1}^{|U|} w_j a_j) \Big| \Big\} \\
&= \sum_{i=1}^{|U|} \sum_{\mathbf{x}} \Pr(\mathbf{x}) \Big| f(\sum_{j=i}^{n} w_j x_j + \sum_{j=1}^{i-1} w_j a_j) - f(\sum_{j=i+1}^{n} w_j x_j + \sum_{j=1}^{i} w_j a_j) \Big|
\end{aligned}$$

Now let B_i be x_i's blocking set and let $W \stackrel{\text{def}}{=} X - (B_i \cup \{x_i\})$. Then, we can write:

$$\begin{aligned}
err(c, c_U) &\leq \sum_{i=1}^{|U|} \sum_{\mathbf{x}_W} \sum_{\mathbf{b}_i} \sum_{a'_i \in \{a_i, \overline{a_i}\}} \Pr(\mathbf{x}_W | B_i = \mathbf{b}_i) \Pr(B_i = \mathbf{b}_i | x_i = a'_i) \Pr(x_i = a'_i) \\
&\quad \times \Big| f(\sum_{j=i+1}^{n} w_j x_j + w_i x_i + \sum_{j=1}^{i-1} w_j a_j) - f(\sum_{j=i+1}^{n} w_j x_j + w_i a_i + \sum_{j=1}^{i-1} w_j a_j) \Big| \\
&= \sum_{i=1}^{|U|} \Pr(x_i = \overline{a_i}) \sum_{\mathbf{b}_i} \Pr(B_i = \mathbf{b}_i | x_i = \overline{a_i}) \\
&\quad \times \text{Binf}(x_i | B_i = \mathbf{b}_i, x_{i-1} = a_{i-1}, \cdots, x_1 = a_1) \qquad (2.14)
\end{aligned}$$

where the last line is obtained from the definition of the block influence.

Now let's define:

$$|\text{Binf}(x_i | x_{i-1} = a_{i-1}, \cdots, x_1 = a_1)| \stackrel{\text{def}}{=} \max_{\mathbf{b}_i} |\text{Binf}(x_i | B_i = \mathbf{b}_i, x_{i-1} = a_{i-1}, \cdots, x_1 = a_1)| ,$$

then:

$$\begin{aligned} err(c, c_U) &\leq \sum_{i=1}^{|U|} \Pr(x_i = \overline{a_i}) \left|\mathrm{Binf}(x_i | x_{i-1} = a_{i-1}, \cdots, x_1 = a_1)\right| \times \\ &\quad \sum_{\mathbf{b}_i} \Pr(B_i = \mathbf{b}_i | x_i = \overline{a_i}) \\ &= \Pr(x_i = \overline{a_i}) \left|\mathrm{Binf}(x_i | x_{i-1} = a_{i-1}, \cdots, x_1 = a_1)\right| . \end{aligned} \tag{2.15}$$

Since there always exists a set of values $\{a_{i-1}, a_{i-2}, \cdots, a_1\}$ such that:

$$\left|\mathrm{Binf}(x_i | x_{i-1} = a_{i-1}, \cdots, x_1 = a_1)\right| \leq \left|\mathrm{Binf}(x_i)\right| . \tag{2.16}$$

For such a setting of the a_is, we will find:

$$err(c, c_U) \leq \sum_{i=1}^{|U|} \Pr(x_i = \overline{a_i}) \left|\mathrm{Binf}(x_i)\right| \tag{2.17}$$

Therefore, as we mentioned, the lemma holds for $\theta = \sum_{i=1}^{|U|} w_i a_i$. □

After discovering the weights, the hypothesis p-concept h returned by the learner will simply be the table look-up of the estimated probabilities of observing a positive classification given that $\sum_{i=1}^{n} w_i x_i = s$ for all s values that are observed with sufficient probability (the hypothesis can output any value for the values of s that are observed very rarely). We thus have the following learning algorithm for stochastic perceptrons.

Algorithm LearnSP$(n, \epsilon, \delta, \{B_i\}_{i=1}^{n})$

1. Call $m = 128 \left(\frac{2n}{3\epsilon}\right)^{2k+4} \ln\left(\frac{32n}{\delta}\right)$ training examples (where $k = max_i|B_i|$).

2. Compute $\hat{\Pr}(x_i = +1)$ for each variable x_i. Neglect x_i whenever we have $\hat{\Pr}(x_i = +1) < 3\epsilon/(4n)$ or $\hat{\Pr}(x_i = +1) > 1 - 3\epsilon/(4n)$.

3. For each variable x_i and for each of its blocking vector value $\mathbf{b}_i$, compute $\hat{\text{Binf}}(x_i|\mathbf{b}_i)$. Let $\mathbf{b}_i^*$ be the value of $\mathbf{b}_i$ for which $|\hat{\text{Binf}}(x_i|\mathbf{b}_i)|$ is the largest. Let $\hat{\text{Binf}}(x_i) = \hat{\text{Binf}}(x_i|\mathbf{b}_i^*)$.

4. For each variable x_i:

   (a) Let $w_i = +1$ whenever $\hat{\text{Binf}}(x_i) > 3\epsilon/(4n)$.

   (b) Let $w_i = -1$ whenever $\hat{\text{Binf}}(x_i) < -3\epsilon/(4n)$.

   (c) Otherwise let $w_i = 0$

5. Compute $\hat{\Pr}(\sum_{i=1}^n w_i x_i = s)$ for $s = -n, \ldots + n$.

6. Return the hypothesis p-concept h formed by the table look-up:

$$h(\mathbf{x}) \;=\; h'(s) \;=\; \widehat{\Pr}\left(\sigma = 1 \,\middle|\, \sum_{i=1}^{n} w_i x_i = s\right)$$

   for all s for which $\hat{\Pr}(\sum_{i=1}^n w_i x_i = s) > \epsilon/(8n+8)$. For the other s values, let $h'(s) = 0$ (or any other value).

Theorem 2.1 *Algorithm* **LearnSP** *PAC learns the class of stochastic perceptrons on $\mathcal{I}^n$ with monotonic activation functions and weights $w_i \in \{-1, 0, +1\}$ under any k-blocking distribution (when a blocking set for each variable is known). The number of examples required is $m = 128\left(\frac{2n}{3\epsilon}\right)^{2k+4} \ln\left(\frac{32n}{\delta}\right)$ (and the time needed is $O(n \times m)$) for the returned hypothesis to make error at most ϵ with confidence at least $1 - \delta$.*

Proof: From Hoeffding's inequality (lemma 2.3) we can show that this sample size is sufficient to ensure that:

- $\left|\hat{\Pr}(x_i = +1) - \Pr(x_i = +1)\right| < \epsilon/(4n)$ with confidence at least $1 - \delta/(4n)$

- $\left|\hat{\text{Binf}}(x_i) - \text{Binf}(x_i)\right| < \epsilon/(4n)$ with confidence at least $1 - \delta/(4n)$
- $\left|\hat{\Pr}(\sum_{i=1}^{n} w_i x_i = s) - \Pr(\sum_{i=1}^{n} w_i x_i = s)\right| < \epsilon^2/[64(n+1)]$ with confidence at least $1 - \delta/(4n+4)$
- $\left|\widehat{\Pr}(\sigma = 1 \,|\sum_{i=1}^{n} w_i x_i = s) - \Pr(\sigma = 1 \,|\sum_{i=1}^{n} w_i x_i = s)\right| < \epsilon/4$ with confidence at least $1 - \delta/4$

If we choose m in order to satisfy the second inequality with the required confidence, all the inequalities above will be satisfied (with the required confidence). That value of m can be found by using Hoeffding's inequality (see appendix B for details). □

2.5 Conclusion

We have presented a simple polynomial time algorithm for PAC learning stochastic perceptrons under the family of k-blocking distributions. This family of distributions represent an important step beyond the case where each input variable is statistically independent (*i.e.* product distributions) since the 2k-blocking family distribution contains all the Markov distributions of order k. However, we restricted ourselves to the case where the learner had information about the blocking set of each variable.

Chapter 3

A Statistical Method for Learning Nonoverlapping Neural Networks

In this chapter, we present a statistical method that PAC learns, under product distributions, the class of nonoverlapping perceptron networks of arbitrary depth, with weights taken from $\{-1, 0 + 1\}$, and arbitrary thresholds.

3.1 Introduction

From a theoretical perspective, it is well known that efficient learning of non simple neural network classes is possible only when either the learner is able to use membership queries or when the distribution that generates the input examples is restricted to some well defined class. Following several positive learnability results on different classes of *read-once* Boolean formulas, a membership query algorithm has been proposed [12] for learning the class of *nonoverlapping* perceptron networks (here and henceforth NPN). These networks, also known as μ-perceptron networks or read-once formulas over a weighted threshold basis, are loop-free neural nets in which each node has only one outgoing weight. If membership queries are not permitted (as we assume throughout

this thesis), learning this class becomes intractable [17] under arbitrary input distributions. Under the *uniform* distribution, however, a PAC learning algorithm has been proposed [10] for a quite restricted subclass called *generalized μ-perceptron decision lists*. Inspired by [29], we have found more recently [23] an algorithm that exactly learns the class of constant depth μ-perceptron networks under *product distributions*, *i.e.* distributions in which the setting of each input variable is chosen independently of the other variables. In the above work however, "weakly relevant" variables such as those that occur deep in the target network had to be maintained in the hypothesis network to be assured to find the correct network connectivity. As a consequence, learning from a polynomial size sample could be achieved only for fixed depth networks. Here, we show how to construct an hypothesis network only from the "relevant" variables in order to PAC learn (in poly-time) the class of NPN of arbitrary depth under product distributions. Despite the fact that we still exploit the *strong unimodality* [16] property of sums of independent random variables to find the network connectivity, the statistical tests used for this purpose are different and leads to a simpler learning algorithm.

Because of its statistical nature [18], the proposed algorithm can tolerate a classification noise rate η up to the information theoretic limit of $\eta = 1/2$.

3.2 Preliminary definitions

The NPN function class is the set of all Boolean functions of the Boolean domain $\{0,1\}^n$ that can be represented as a loop-free network of perceptrons where each node (including input units) has only one outgoing weight (see fig 3.1 below for an example NPN function). Let $X = \{x_1, x_2, \cdots, x_n\}$ be the set of n input variables and $\mathbf{x} \in \{0,1\}^n$

be some assignment of these n variables, we denote by $\mathbf{x}_V$ the restriction of assignment $\mathbf{x}$ on the variables in $V \subseteq X$. A *perceptron* g on V is defined by a vector of $v = |V|$ weights w_i and a single threshold θ. For any $\mathbf{x}_V \in \{0,1\}^v$, the output of $g(\mathbf{x}_V)$ is 1 whenever $\sum_{i \in V} w_i x_i > \theta$ and 0 otherwise. We restrict ourselves to the case where each $w_i \in \{-1, 0, +1\}$ but the thresholds are arbitrary so that, without loss of generality (w.l.o.g.), $\theta \in \{-v-1, -v, \cdots, +v\}$.

Each perceptron can have variables and/or other perceptrons as inputs. Hence, a *node* will denote either a variable or a perceptron. The output unit of a network will often be referred as the *root node*. We say that a node is a *child* of the *parent* perceptron g if it is an immediate input to perceptron g. Children of the same perceptron are called *siblings*. A perceptron is said to be a *bottom level perceptron* if all its children are variables.

We consider that each example $\mathbf{x}$ is generated by an unknown product distribution D on $\{0,1\}^n$ so that all probabilities are defined with respect to D. If f denotes the target NPN and h denotes an hypothesis NPN returned by the learner, the error that h makes with f, denoted by $\mathrm{err}(h, f)$, is defined as $\Pr\{h(\mathbf{x}) \neq f(\mathbf{x})\}$.

The learning algorithm will satisfy the *PAC learning criteria* [30] which, in our case, can be stated in the following way. For any target NPN function f and for any ϵ, δ, n given to the learner, the probability will be at least $1-\delta$ that the returned hypothesis h satisfies $\mathrm{err}(h, f) < \epsilon$. The number of examples needed by the algorithm to satisfy this criteria and it's running time will be both bounded by a polynomial in $(n, 1/\epsilon, 1/\delta)$.

3.3 Finding the weights of the relevant variables

To construct an hypothesis which achieves PAC learnability, the learner needs only the variables which are *significantly relevant*. Variables which are (almost) irrelevant can be classified as being either *sticky* or *weakly influential*.

Following Schapire [29], a variable x_i is said to be ϵ_S-sticky if there exists $a_i \in \{0,1\}$ such that $\Pr(x_i = a_i) > 1 - \epsilon_S$. It follows that if U is a set of ϵ_S-sticky variables, then there exists a function f_U which is obtained from f by replacing the reading of each ϵ_S-sticky variable x_i by a constant value a_i such that $\mathrm{err}(f_U, f) \leq \sum_{i \in U} \Pr(x_i \neq a_i) < |U|\epsilon_S$. Hence, after choosing a value for ϵ_S which is small enough, the learner will build his hypothesis only from the non sticky variables.

To find the weights, note that we can assume, w.l.o.g., that only input variables have a negative outgoing weight. Indeed, if a perceptron g has a -1 outgoing weight, we can replace it by a perceptron which has all its incoming weights negated and a $+1$ outgoing weight; this leaves the computation by f unchanged when we add $+1$ to the threshold of g's parent. In this manner, all -1 weights are pushed to the input variables. To identify the weight w_i that springs out of each input variable x_i we will estimate its *influence*, defined as:

$$\mathrm{Infl}(x_i) \stackrel{\text{def}}{=} \Pr(f = 1 | x_i = 1) - \Pr(f = 1 | x_i = 0)$$

We will also use, $\mathrm{Infl}_g(x_i)$ to denote the influence of x_i on the subformula of f which is rooted at perceptron g. Also, $\mathrm{Infl}(x_i | x_j = a)$ will denote the influence of x_i given that variable x_j is fixed to value a. To find each w_i we use:

Lemma 3.1 *Let f be any NPN with weights taken from $\{-1, 0, +1\}$ and arbitrary thresholds. Let D be any product distribution on $\{0,1\}^n$. Then for any input variable*

x_i with outgoing weight w_i:

$$\text{Infl}(x_i) \begin{cases} \geq 0 & \text{if } w_i = +1 \\ = 0 & \text{if } w_i = 0 \\ \leq 0 & \text{if } w_i = -1 \end{cases}$$

Proof idea: The proof follows directly from the definition of Infl(·) and of μ-perceptron networks (under product distributions) when all the -1 weights are outgoing from the input variables only. □

Lemma 3.1 tells us that the weight outgoing from x_i is obtained from the sign of $\text{Infl}(x_i)$. In the advent that $|\text{Infl}(x_i)|$ is too small to estimate the sign of $\text{Infl}(x_i)$ from a poly-size sample, the next lemma will tell us that we can ignore such x_i without making much error.

Lemma 3.2 *Let the target NPN function f be such that there does not exists a value $a \in \{0,1\}$ such that $\Pr(f = a) > 1 - \epsilon_f$. Let $U \subseteq X$ be a set of u variables such that for all $x_i \in U$, $|\text{Infl}(x_i)| < \epsilon_I$. Let f_U be the function obtained from f by replacing the reading of each $x_i \in U$ by the reading of a constant value $a_i \in \{0,1\}$. Then, for any set of values $\{a_1, a_2, \cdots, a_u\}$, we have:*

$$\text{err}(f_U, f) \leq 2u\epsilon_I$$

whenever $\epsilon_I \leq \epsilon_f/(8n)$.

Proof: see appendix C. □

Given a target NPN f, lemma 3.2 tells us that there exists another (hypothesis) NPN h of equivalent connectivity that depends only on variables x_i having $|\text{Infl}(x_i)| \geq \epsilon_I$ and which makes a small error with the target NPN f.

After eliminating sticky and weakly influential variables, we now transform the target function f into its *positive form* simply by changing x_i to $1 - x_i$ (and adding $+1$ to the threshold of x_i's parent) whenever $w_i = -1$. The target function f expressed in terms of the remaining variables is now stochastic. We will refer to these remaining variables as the *relevant* variables.

3.4 Overview on how to construct the skeleton

To describe the algorithm used by the learner for building a skeleton of the hypothesis function h, we need to introduce more definitions.

Recall that a *node* refers to either a variable or a perceptron. The *depth* of a node is defined as the number of perceptrons (including the parent of the node and the root node) on the path from the parent of the node to the root. The perceptrons on this path are called the *ancestors* of that node. The *descendants* of perceptron g, denoted by $\text{desc}(g)$, is the set of nodes that have perceptron g as an ancestor. The depth of a network is defined as the depth of the deepest variable in the net.

The *least common ancestor* of a set V of variables, denoted by $\text{lca}(V)$, is defined as the deepest perceptron ancestor which is common to every variable in V. A set V of three or more variables is said to be *meeting*, *iff* there exists a perceptron g such that for any triple $\{x_i, x_j, x_k\} \in V$ we have: $\text{lca}(x_i, x_j) = \text{lca}(x_i, x_k) = \text{lca}(x_j, x_k) = g$. If there does not exist a perceptron g having this property, then V is said to be *not meeting*. Any set containing 2 variables is said to be meeting.

A meeting set V is said to be *complete* with respect to a reference set $W \supseteq V$ *iff* there does not exist $x_k \in W - V$ such that $V \cup \{x_k\}$ is meeting. A complete meeting

set V w.r.t. W is said to be a *complete meeting set at the lowest level* (CMSLL) w.r.t. W *iff* there does not exist another complete meeting set $U \neq V$ w.r.t. the same W such that lca(U) is a descendant of lca(V).

A perceptron g can happen to be either an OR gate or an AND gate. Such a perceptron is referred to as a *G-perceptron.* Any perceptron which is not a G-perceptron is referred to as an *NG-perceptron.*[1] A CMSLL that meets at a G-perceptron is called a G-CMSLL and a CMSLL that meets at an NG-perceptron is called an NG-CMSLL.

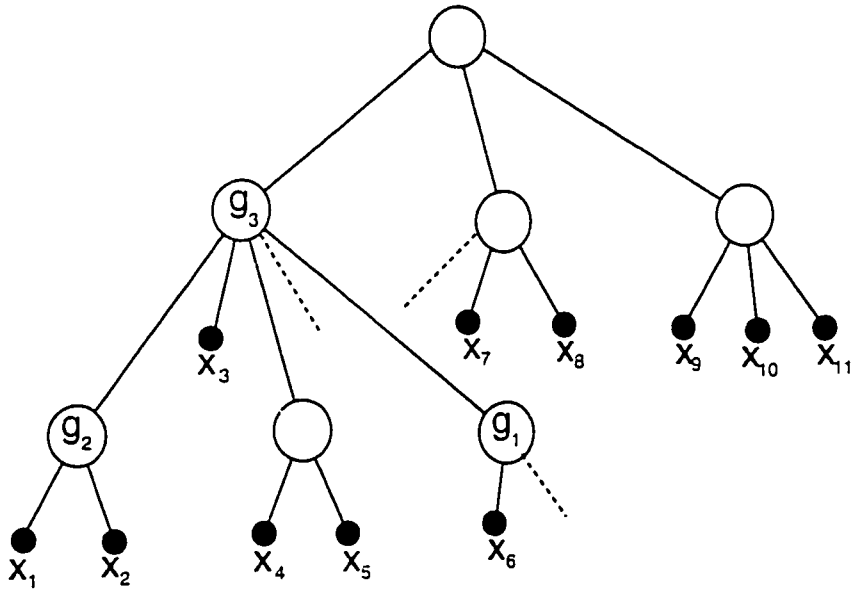

Figure 3.1: An example NPN used to illustrate the various definitions (see text). The dashed lines indicate variables which have been removed.

Figure 3.1 illustrates these various definitions. Set $\{x_2, x_3, x_4\}$ and set $\{x_9, x_{10}, x_{11}\}$ are two examples of sets which are meeting. However, set $\{x_1, x_2, x_3\}$ is not meeting. The sole purpose of the various statistical tests that we introduce below is to permit the learner to identify all the CMSLL w.r.t. any given set W of variables. If W consists of all the relevant variables, then each CMSLL w.r.t. W identifies a bottom level percep-

[1]The letters G and NG, respectively, are abbreviations for gate and for non-gate.

tron. Referring again to Fig. 3.1, the following sets $\{x_1, x_2\}, \{x_4, x_5\}, \{x_7, x_8\}, \{x_9, x_{10}, x_{11}\}$ are the only ones which are a CMSLL w.r.t. W when W consist of all the relevant variables. The learner will associate each CMSLL with a bottom level perceptron. But note that it has missed the bottom level perceptron g_1 since it contains only one relevant child, x_6, which we call a *singleton*. This is of no consequence to a learner whose task is not to model the probabilistic behavior induced by ignoring "weekly relevant" variables but to try to find the best deterministic hypothesis. For this goal, it is better to connect a singleton to his next ancestor (*i.e.* here to connect x_6 to g_3). We will describe later other examples where the learner does not identify the exact skeleton of the target function but an equivalent one with respect to the relevant variables.

Let us now update W by removing all the variables, except one that we call a *group representative*, per each CMSLL so that W becomes $\{x_1, x_3, x_4, x_6, x_7, x_9\}$ in Fig. 3.1. Then the only CMSLL w.r.t this new W is $\{x_1, x_3, x_4, x_6\}$ which the learner will associate with the perceptron g_3, parent of x_3 and (now) x_6 and of the bottom level perceptrons that were associated previously with $\{x_1, x_2\}$ and $\{x_4, x_5\}$.

Finally, after removing from W every variable (except one representative) of this CMSLL, we are left with $W = \{x_1, x_7, x_9\}$ which is a CMSLL w.r.t. W. This last CMSLL which equals to W identifies the root and terminates the algorithm for building the skeleton. The last task of the learner should then be to compute good threshold values for each identified perceptron, but, as we will describe later, we will instead provide for each of them an activation function that provides a good model of probability of the underlying perceptron. Since $w_i \in \{-1, 0, +1\}$, each of these perceptrons has at least two children. This implies that h has at most $n - 1$ perceptrons.

Another strategy, used in [23], to construct the skeleton would have been to com-

pute the threshold of a perceptron immediately after its discovery and then use this perceptron as a new "meta-variable" (that replace its children) to discover its parent. But this strategy has no guarantee of success here. Indeed, in contrast with [23], we are now forced to build an hypothesis on a possibly smaller set of (relevant) input variables. In doing so, each perceptron g' of the hypothesis will possibly be a noisy approximate of the "true" perceptron g of the target function. The statistical tests that we use below will provably construct the correct skeleton only when using them on noiseless variables. This is why the learner uses only the true input variables in the statistical tests to build the skeleton.

3.5 Main statistical tests for finding CMSLLs

We now describe the statistical tests that the learner will use to identify all the CMSLLs w.r.t. any given set of variables. The key quantity to estimate is what we define as the *coinfluence* of a triple of variables:

$$\begin{aligned} C_{i,j}^{k} \stackrel{\text{def}}{=} & \frac{\mathrm{Infl}(x_j|x_i=1,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=0)} - \frac{\mathrm{Infl}(x_j|x_i=1,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=1)} \\ + & \frac{\mathrm{Infl}(x_j|x_i=0,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=1)} - \frac{\mathrm{Infl}(x_j|x_i=0,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=0)} \end{aligned}$$

We will see below that the denominators $\mathrm{Infl}(x_j|x_k=a)$ may vanish whenever x_k is a child of a G-perceptron. Hence, before estimating $C_{i,j}^{k}$, we must make sure that x_j is not a descendant of the parent of x_k when this parent happens to be a G-perceptron.

Before stating the relevant lemmas we recall the strong unimodality property of sums of independent random variables:

Lemma 3.3 ([23]) *Let* $\{x_1, x_2 \cdots, x_v\}$ *be* v *independent random Boolean variables, each with* $\Pr(x_i = 1) = q_i$ *and let* $S \stackrel{\text{def}}{=} \sum_{i=1}^{v} x_i$. *Then for any* $\{q_1 \cdots, q_v\}$ *and any* $k \in \{1, \cdots, v\}$:

$$\frac{\Pr(S = k-1)}{\Pr(S = k)} - \frac{\Pr(S = k-2)}{\Pr(S = k-1)} \geq \frac{1}{v < \alpha >_v}$$

where $\alpha_i \stackrel{\text{def}}{=} q_i/(1-q_i)$ *and* $< \alpha >_v \stackrel{\text{def}}{=} \sum_{i=1}^{v} \alpha_i / v$.

moreover,

$$\frac{\Pr(S = k)}{\Pr(S = k-1)} \geq \frac{1}{v < 1/\alpha >_v}$$

where $< 1/\alpha >_v \stackrel{\text{def}}{=} \sum_{i=1}^{v} \alpha_i^{-1} / v$.

proof: see appendix D. □

Hence, we define the *strong unimodal gap* γ_n to be:

$$\gamma_n \stackrel{\text{def}}{=} \min \left\{ \frac{1}{n < \alpha >_n}, \frac{1}{n < 1/\alpha >_n} \right\}.$$

To identify the children of G-perceptrons we will use the following coinfluences of pair of variables:

$$A^1_{j,i} \stackrel{\text{def}}{=} \frac{\text{Infl}(x_j | x_i = 1)}{\text{Infl}(x_j)} \quad \text{and} \quad A^0_{j,i} \stackrel{\text{def}}{=} \frac{\text{Infl}(x_j | x_i = 0)}{\text{Infl}(x_j)}.$$

Lemma 3.4 *If a variable* x_i *is a child of an OR gate then there exists* x_j *such that* $A^1_{j,i} = 0$. *Otherwise,* $A^1_{j,i} > \gamma_n$ *for all* $x_j \neq x_i$.

This lemma is valid when we replace OR by AND if $A^1_{j,i}$ *is replaced by* $A^0_{j,i}$.

proof: see appendix E. □

Hence, by the presence of the strong unimodal gap in the coinfluences, we can identify the children of G-perceptrons (*i.e.* OR/AND gates) from $A^a_{j,i}$ ($a \in \{0, 1\}$).

However, the learning algorithm has access only to the empirical estimates $\hat{A}^a_{j,i}$ obtained from the training examples. Nevertheless, the well known bounds of Chernoff and Hoeffding [11] tell us that a poly-size sample is sufficient in order to have $\hat{A}^a_{j,i}$ close enough to $A^a_{j,i}$ with high probability.

3.6 Finding children of OR/AND gates

By exploiting lemma 3.4, the algorithm **FindOR** (see next) will identify all variables which are children of some OR gates. It will be used only once, at the beginning of the building process for the skeleton to identify groups of variables which cannot be used jointly for estimating coinfluences of triples of variables.

Following the definition of a CMSLL, a set V of siblings to some perceptron is said to be *complete* w.r.t. the set of all relevant variables W if there does not exist $x_k \in W - V$ such that $V \cup \{x_k\}$ is a set of siblings. The algorithm **FindOR** will output all the complete sets of siblings of OR gates. Each of these sets will be classified either as a *complete set of siblings of a bottom level OR gate* or as a *complete set of siblings of a non-bottom level OR gate* depending on whether or not the OR gate is bottom level. In the latter case, a set will contain only one variable whenever all its siblings are perceptrons. The algorithm **FindAND** will perform the same task as **FindOR** but for AND gates and by using $A^0_{j,i}$ instead of $A^1_{j,i}$.

Algorithm FindOR

Input: The set W of all relevant variables; $|W|$ must be 2 or more.

Output: A set $T_b = \{B_1, B_2, \cdots\}$, a set $T_{nb} = \{(N_1, D_1), (N_2, D_2), \cdots\}$. Each B_i is a complete set of siblings of a bottom level OR gate. Each N_i is a complete set of

siblings of a non bottom level OR gate. Each D_i is the set of variables, excluding those in N_i, which are descendants of the parent of N_i.

Initialization: Set $T_b = \emptyset$, $T_{nb} = \emptyset$.

1. **If** $|W| \leq 1$, **then** return; **else** $P = W$.
2. **If** $P = \emptyset$, **then** return; **else** Choose any $x_i \in P$.
3. Set $V_d = \{x_j (\in W) \neq x_i : \hat{A}^1_{j,i} \leq \gamma_n/2\}$
   and set $V_{gc} = \{x_j \in V_d : \hat{A}^1_{i,j} > \gamma_n/2\}$
4. **If** $|V_d| \geq 1$, **then** set $V_d = V_d \cup \{x_i\}$;
   else Set $P = P - \{x_i\}$, and goto 2.
5. **If** $|V_{gc}| \leq 1$, **then** { Set $T_b = T_b \cup \{V_d\}$, set $P = P - V_d$, goto 2};
   else { Set $V_c = V_d - V_{gc}, T_{nb} = T_{nb} \cup \{(V_c, V_{gc})\}, P = P - V_c$, goto 2 }.

To make this algorithm more transparent, note that V_c, V_d, and V_{gc} refer respectively to the set of children, the set of descendants, and the set of grandchildren of the parent of a given variable x_i. We have that $V_d = V_c \cup V_{gc}$, $V_c \cap V_{gc} = \emptyset$ and do not discriminate between grandchildren and great grandchildren.

Note that when $|V_{gc}| = 1$, we consider, in step 5, that V_d is a complete set of siblings of bottom level gate. Indeed, in this case V_{gc} contains a singleton and, as explained before, it is better for the learner to connect it to the parent of x_i. As we will see, the case where V_{gc} contains two or more singletons is handle correctly by the next algorithms **FindNG-CMSLL** and **FindG-CMSLL**.

3.7 Finding NG-CMSLLs

After running **FindOR** and **FindAND** on the set W of all relevant variables, we will update W in the same way as we have described for the CMSLLs. For each complete set of siblings of a bottom level G-perceptron, we are going to delete from W and each D_i all the variables except one (the group representative). Furthermore, we are going to remove from W all the variables which are in some set N_i (a complete set of siblings of a non bottom level G-perceptron). Then, W will have the property that no subset $U = \{x_j, x_k\}$ of two variables exists in W such that x_k is a child of a G-perceptron and x_j is a descendent of the parent of x_k. Hence, we are going to use the following definition:

Definition 3.1 C-safe set

A C-safe set V is a set that does not have any subset $U = \{x_j, x_k\} \subseteq V$ of two variables such that x_k is a child of a G-perceptron and x_j is a descendent of the parent of x_k.

From Definition 3.1, a C-safe set V has a property that no denominator of $C_{i,j}^k$ on any triple of variables $\{x_i, x_j, x_k\}$ will vanish when $\{x_j, x_k\} \in V$. In other words, it is safe to estimate $C_{i,j}^k$ on any triple of variables $\{x_i, x_j, x_k\}$ where x_j and x_k are members of a C-safe set V. Hence, the updated set W, obtained after running **FindOR** and **FindAND** on the original set of all relevant variables, will be a C-safe set. For such a C-safe set W, here are the key properties of $C_{i,j}^k$ that the learner is going to exploit.

Lemma 3.5 *Let $V = \{x_i, x_j, x_k\}$ be any C-safe set of three relevant variables. Let $\omega \stackrel{\text{def}}{=} \epsilon_I \gamma_n^2$. Then, the following properties hold:*

1. $C^k_{i,j} > \omega$ *if* $\{x_i, x_j, x_k\}$ *are siblings of an NG-perceptron or meet at an NG-perceptron.*

2. $C^k_{i,j} = 0$ *if* $x_k \notin \text{desc}(\text{lca}(x_i, x_j))$, *or* $x_i \notin \text{desc}(\text{lca}(x_j, x_k))$, *or* $\{x_i, x_j, x_k\}$ *meet at a G-perceptron.*

3. $C^k_{i,j} > \omega$ *if* $x_j \notin \text{desc}(\text{lca}(x_i, x_k))$ *and* $\text{lca}(x_i, x_j) = \text{lca}(x_j, x_k)$ *is a G-perceptron.*

Proof: See appendix F. □

The next lemma exploits the properties of lemma 3.5 and yields tools to the learner so that it can perform the crucial task of deciding whether or not a set V of three or more relevant variables is an NG-CMSLL w.r.t. a C-safe set W. The NG-CMSLLs of two relevant variables will be discussed in the next section.

Lemma 3.6 *Let W be a C-safe set of at least three relevant variables. Then, a set $V \subseteq W$, containing at least three variables, is an NG-CMSLL w.r.t. W iff the following conditions are satisfied simultaneously:*

1. $C^k_{i,j} \geq \omega$ *for all* $\{x_i, x_j, x_k\} \subseteq V$

2. $C^l_{i,j} = 0$ *for all* $\{x_i, x_j\} \subset V$ *and* $x_l \in W - V$.

Proof: Proof directly follows from the first two properties of Lemma 3.5. □

By exploiting lemma 3.6, the algorithm **FindNG-CMSLL** (next) will find all NG-CMSLLs of three or more relevant variables w.r.t. a C-safe set W. The algorithm will terminate when all NG-CMSLLs in W are found. As usual, in the algorithm, $\hat{C}^k_{i,j}$ denotes the empirical estimate of $C^k_{i,j}$.

Algorithm FindNG-CMSLL

Input: A C-safe set W of at least three relevant variables.

Output: The set $T = \{V_1, V_2, \cdots\}$ where each V_i is an NG-CMSLL w.r.t. W.

Initialization: Set $T = \emptyset$.

1. Let P be the set of all pairs $\{x_i, x_j\} \subseteq W$.
2. **If** $P = \emptyset$, **then** return;
   else {Choose $V = \{x_i, x_j\} \in P$ and set $P = P - \{V\}$}.
3. Set $V = V \cup \{x_k \in W - V : \hat{C}^k_{i,j} > \omega/2\}$.
4. **If** $\{|V| = 2$ OR $\hat{C}^k_{i,j} \leq \omega/2$ for any permutation of $\{x_i, x_j, x_k\} \subseteq V\}$, **then** {goto 2};
   else { Set $T = T \cup \{V\}$, set $W = W - V$, and goto 1}.

3.8 Finding G-CMSLLs

To find the G-CMSLLs, we will first need to identify all the NG-CMSLLs. To do this, we need to run the algorithm **FindNG-CMSLL** on the updated set W until no NG-CMSLLs can be found. After each run, we update the set W in the usual way, *i.e.* by keeping only one group representative within each of the NG-CMSLLs. If the updated W contains only two elements, they must meet at the root and the skeleton is found. Hence, throughout this section we suppose that the updated set W, inside which we are going to look for G-CMSLLs, contains three or more variables.

Lemma 3.5 states that a G-CMSLL will give $C_{i,j}^k = 0$ for all of its triples of variables. However, it also leaves the possibility that we have $C_{i,j}^k = 0$ for all permutations of a triple of variables which does not meet (when the lca of this triple is not a G-perceptron). Unfortunately, we have identified such cases. Hence, the fact that $C_{i,j}^k = 0$ for all permutations of a triple of variables does not determine by itself that this triple of variables meet at a G-perceptron.

To motivate the additional lemmas and tests that will be needed to find the G-CMSLLs, consider the different cases illustrated on Fig. 3.2.

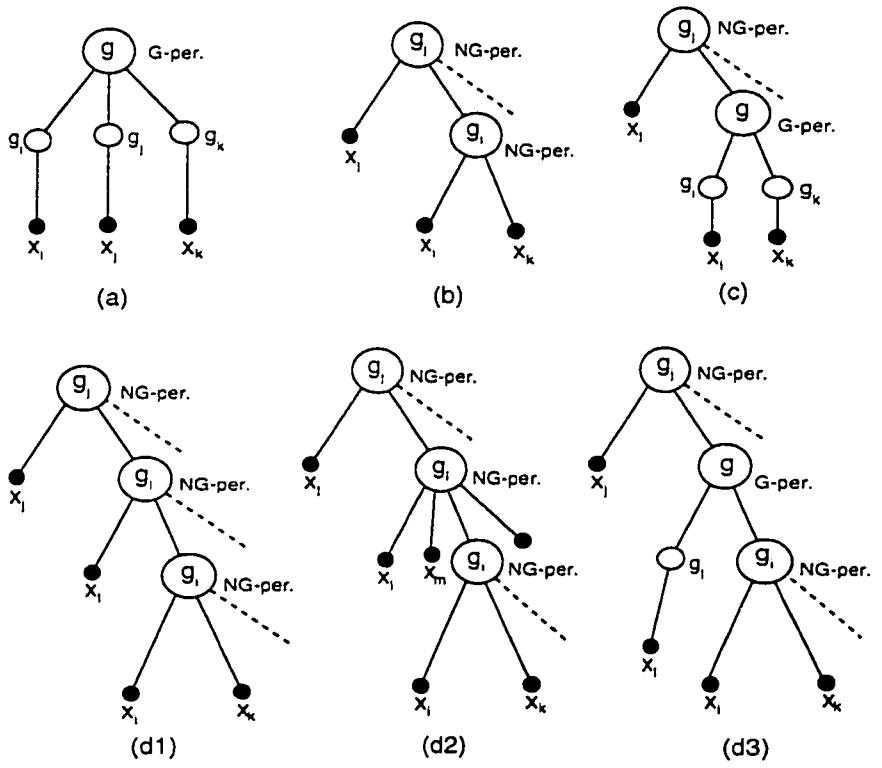

Figure 3.2: (a) is the case where we always have $C_{i,j}^k = 0$ (Lemma 3.5). (b), (c) and (d) are the cases where we *might* have $C_{i,j}^k = 0$ (see the text). The dashed lines indicate variables which have been removed.

In case (a), x_i, x_j, and x_k are group representatives of perceptrons g_i, g_j, and g_k,

respectively. These variables meet at a G-perceptron and Lemma 3.5 states that we have $C_{i,j}^{k} = 0$ for all permutations of these variables. However, we might also have this property for all the other cases of Fig. 3.2 where these variables do not meet.

Consider case (b). Since all the variables that we are considering in this section must belong to a C-safe set, both g_i and g_j must be NG-perceptrons. This implies that some (weakly relevant) variables were removed from both g_i and g_j (otherwise, they would be G-perceptrons). However, in this case, Lemma 3.2 tells us that g_i and g_j can be replaced in the hypothesis by either OR or AND gates without introducing much error. In turn, this implies that this whole structure for case (b) can be replaced by a G-perceptron (either an OR or an AND gate) of $\{x_i, x_j, x_k\}$ without introducing much error. Hence, we say that in such case, the learner *can consider* (without introducing much error) that $\{x_i, x_j, x_k\}$ is a G-CMSLL. Therefore, this case does not, by itself, motivate the introduction of another test.

Considering again case (b), note that there cannot exist in W another (relevant) variable, say x_l, which would be a child of g_i since, otherwise, $\{x_i, x_k, x_l\}$ would be an NG-CMSLL meeting at a NG-perceptron and these sets have been removed from W. However, g_j could have another relevant child, call it x_l, but then we would have (from Lemma 3.5) $C_{i,j}^{l} > \omega$. Hence, for this reason, we state that:

A set V can be considered as a G-CMSLL w.r.t. a C-safe set W that contains no NG-CMSLL of three or more relevant variables only if $C_{i,j}^{l} = 0$ for all $\{x_i, x_j\} \subset V$ and $x_l \in W - \{x_i, x_j\}$.

From Lemma 3.5, it is clear that any CMSLL w.r.t. a C-safe set W that contains no NG-CMSLL of 3 or more relevant variables satisfies this condition. Hence, all such CMSLLs can be considered as a G-CMSLL in the hypothesis.

To determine if this condition is *sufficient* for a set V to able to be considered as a G-CMSLL, let us consider the situations illustrated in cases (d1), (d2), and (d3) of Fig. 3.2. These are similar to the case (b) except that g_i now feeds g_j through another perceptron g_l. Note that g_l can be an NG-perceptron which is fed by one variable x_l (beside x_i and x_k through g_i) and some removed variables. This is illustrated in the case (d1). As illustrated in the case (d2), g_l can also be an NG-perceptron of three or more variables. Also g_l can be a G-perceptron as in the case (d3). But in the latter case, x_l cannot be a child of g_l since algorithm **FindOR/AND** has removed such children. Instead, x_l can be a singleton or a group representative that has remained at this stage. Note that the set $V = \{x_i, x_j, x_k, x_l\}$ may satisfy the condition mentioned above for only the case (d1), then indeed, the set V *can be considered* as a G-CMSLL without introducing much error. We also note that the set $V = \{x_i, x_j, x_k, x_l\}$ cannot satisfy the above condition for the cases (d2) and (d3). For the case (d2), there exists $x_m \in W - V$ which feeds g_l so that $C^m_{i,l} > \omega$, and for case (d3), $C^k_{i,l} > \omega$ from Property 3 of Lemma 3.5. Hence for now, the above condition may seem to be sufficient for a set to be able to be considered as a G-CMSLL.

However, for all three cases (d1), (d2), and (d3), let us now suppose that for the set $V = \{x_i, x_j, x_k\}$, we have $C^l_{i,j} = 0$ for all $\{x_i, x_j\} \subset V$ and $x_l \in W - \{x_i, x_j\}$ and that $C^k_{i,l} > \omega$. By using the above criteria, the learner would incorrectly conclude, in these three cases, that $\{x_i, x_j, x_k\}$ is a G-CMSLL. Therefore, the above condition is clearly not sufficient and we must supply another necessary condition. But for this recall that we have, by hypothesis, for all 3 cases: $C^k_{i,l} > \omega$ while $C^k_{j,l} = 0$. Hence, for this extra condition, we propose that:

A set V can be considered as a G-CMSLL w.r.t. a C-safe set W that contains no NG-CMSLL of three or more relevant variables only if $C^m_{i,l} = C^m_{j,l}$ for all $\{x_i, x_j\} \in V$

and all $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$.

Hence, we can use this condition only when the C-safe set W has at least four variables. The Lemma 3.7 below implies that this condition is always satisfied by CMSLLs w.r.t. a C-safe set that contains no NG-CMSLL of 3 or more relevant variables.

Unfortunately, these two necessary conditions taken together do not constitute a sufficient set. Indeed, consider the case (c) illustrated in Fig. 3.2. Since g is now a G-perceptron and W is a C-safe set, x_i and x_k cannot be children of g. Instead, they can be singletons or group representatives that have remained at this stage. We may find that the two conditions above are satisfied for the set $V = \{x_i, x_j, x_k\}$ *i.e.* $C^l_{i,j} = 0$ for all $\{x_i, x_j\} \subset V$ and $x_l \in W - \{x_i, x_j\}$, and $C^m_{i,l} = C^m_{j,l}$ for all $\{x_i, x_j\} \in V$ and all $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$. Again, Lemma 3.2 tells us that g_j can take either an OR or an AND gate without introducing much error but this is not true for the fixed gate g. Hence, unlike the case (b) of Fig. 3.2, the set $V = \{x_i, x_j, x_k\}$ cannot be considered as a G-CMSLL to either an OR gate or an AND gate. For this reason, we settle for the following definition:

Definition 3.2 G-CMSLL candidate

Let W be a C-safe set of three or more relevant variables in which there is no NG-CMSLL of three or more relevant variables. Then, $V \subseteq W$ is called a G-CMSLL candidate w.r.t. W, iff:

1. $C^l_{i,j} = 0$ *for all* $\{x_i, x_j\} \subseteq V$ *and* $x_l \in W - \{x_i, x_j\}$.
2. *If* $|W| \geq 4$, *then:* $C^m_{i,l} = C^m_{j,l}$ *for all* $\{x_i, x_j\} \subseteq V$ *and all* $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$.
3. *There does not exist* $x_k \in W - V$ *such that (1) and (2) are satisfied for* $V \cup \{x_k\}$.

Furthermore, we will say that the condition (1) fails if, for any $\{x_i, x_j\} \subseteq V$, there exists $x_l \in W - \{x_i, x_j\}$ such that $C^l_{i,j} > \omega$. Also, we will say that the condition (2) fails if, for any $\{x_i, x_j\} \subseteq V$, there exist $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$ such that $|C^m_{i,l} - C^m_{j,l}| > \omega$. Note that we have added a third condition to ensure that a G-CMSLL candidate w.r.t. W is a *complete set* satisfying conditions (1) and (2). Consequently we have the following lemma:

Lemma 3.7 *Let W be a C-safe set of three or more relevant variables in which there is no NG-CMSLL of three or more relevant variables. If V is a CMSLL w.r.t. W, then there exists a set U which is a G-CMSLL candidate w.r.t W such that $V \subseteq U$.*

Proof: See appendix G. □

In other words, Lemma 3.7 says if a set U is a G-CMSLL candidate w.r.t. W, it may or may not be a CMSLL w.r.t. W. However, any CMSLL w.r.t W must be contained in a G-CMSLL candidate w.r.t W.

To identify the different CMSLLs that are contained in a G-CMSLL candidate, we will need to perform another statistical test. Indeed, consider the case (c) of Fig. 3.2 where the set $V = \{x_i, x_j, x_k\}$ is a G-CMSLL candidate whereas only $\{x_i, x_k\} \subset V$ is a G-CMSLL. Since V is a G-CMSLL candidate, then $C^k_{i,j} = 0$. This occurs iff $\text{Infl}(x_j|x_i = 1) = \text{Infl}(x_j|x_i = 0)$ while $\text{Infl}(x_k|x_i = 1) \neq \text{Infl}(x_k|x_i = 0)$ since $\text{lca}(x_i, x_k)$ is a G-perceptron. This suggests to consider the following function:

$$\Gamma_{i,j} \stackrel{\text{def}}{=} \frac{A^1_{i,j} - A^0_{i,j}}{\text{Infl}(x_j)}.$$

Note that $\Gamma_{i,j} = \Gamma_{j,i}$. The above observation leads us to the fact that $C^k_{i,j} = 0$ iff $\Gamma_{i,j} = \Gamma_{k,j} = 0$ while $|\Gamma_{i,k}| > 1$ since $\text{lca}(x_i, x_k)$ is a G-perceptron. (For details, see the proof of the next lemma.)

The next lemma will find out whether or not a G-CMSLL candidate is (or can be considered as) a G-CMSLL and, if not, how to find a G-CMSLL from a G-CMSLL candidate.

Lemma 3.8 *Let W be a C-safe set of three or more relevant variables in which there is no NG-CMSLL of three or more relevant variables. Let V be a G-CMSLL candidate w.r.t. W. Then, we always have one and only one of the following cases for V:*

(a) V is not a G-CMSLL w.r.t. W if there exists a triple $\{x_i, x_j, x_k\} \subseteq V$ such that

$$|\Gamma_{i,j}| > 1 \quad \text{and} \quad |\Gamma_{i,k}| \leq \gamma_n.$$

In this case, $x_j \notin \text{desc}(\text{lca}(\{x_i, x_k\}))$ whenever $\Gamma_{i,k} \neq \Gamma_{i,j} = \Gamma_{j,k}$. Thus x_j does not belong to the same CMSLL that contains $\{x_i, x_k\}$.

(b) V can be considered for the hypothesis as a G-CMSLL w.r.t. W if the condition in (a) does not occur.

Proof: See appendix H . □

When a G-CMSLL V has been found, the gate type of the lca(V) is identified from this corollary (which follows directly from the proof of Lemma 3.8):

Corollary 3.1 *Let V be a G-CMSLL (or being considered as a G-CMSLL in the hypothesis). Let a G-perceptron g, be the* lca(V). *Then:*

i) g is an AND gate if $\Gamma_{i,j} > 1$ for all $\{x_i, x_j\} \in V$.

ii) g is an OR gate if $\Gamma_{i,j} < -1$ for all $\{x_i, x_j\} \in V$.

iii) g can be considered as either an AND or an OR if neither (i) nor (ii) occurs.

By exploiting lemmas 3.7 and 3.8, the algorithm **FindG-CMSLL** (next) is able to find all G-CMSLLs (including NG-CMSLLs that have two variables). But note that the algorithm builds only one G-CMSLL per G-CMSLL candidate at a time. We could have made the algorithm more efficient by constructing several G-CMSLLs per G-CMSLL candidate but this would have made the algorithm more difficult to follow. We have thus decided to present this simpler version. Indeed, this does not change the correctness of the algorithm since the algorithm returns all the G-CMSLLs eventually. As a matter of fact, the algorithm always finds a G-CMSLL candidate from the remaining variables and returns one G-CMSLL in the G-CMSLL candidate until no G-CMSLL candidate remains in the remaining variables. In this way, all the G-CMSLLs will be returned. For example, let the algorithm find U as a G-CMSLL candidate in which V_1 and V_2 are two G-CMSLLs w.r.t. W. At first, the algorithm returns V_1 and removes V_1 from U and from W. Next, the algorithm will find that $U - V_1$ is a G-CMSLL candidate in which V_2 is a G-CMSLL w.r.t. $W - V_1$. So the algorithm returns V_2. Hence, the algorithm will terminate when all G-CMSLLs in W are found. As usual, $\hat{C}^k_{i,j}$ and $\hat{\Gamma}_{i,j}$ denote the empirical estimate of $C^k_{i,j}$ and $\Gamma_{i,j}$ respectively.

Algorithm FindG-CMSLL

Input: A C-safe set W of at least three relevant variables in which there is no NG-CMSLL of three or more relevant variables.

Output: A set $T = \{(V_1, t_1)(V_2, t_2)\cdots\}$ where each V_i is (or can be considered as) a G-CMSLL w.r.t. W and t_i is the gate type associated with the lca(V_i).

Initialization: Set $T = \emptyset$ and set $W' = W$.

1. Let P be the set of all pairs $\{x_i, x_j\} \subseteq W'$.

2. **If** $\{|P| = 0\}$, **then** {return};
   else {Set $V = \{x_i, x_j\} \in P$, and set $P = P - \{V\}$}.

3. Goto 2 **if** this condition is satisfied:
   $\exists\ x_l \in W - V : \hat{C}^l_{i,j} > \omega/2$
   OR
   $\exists\ \{x_l, x_m\} \subseteq W - V : \left|\hat{C}^m_{i,l} - \hat{C}^m_{j,l}\right| > \omega/2.$

4. Let $Q \stackrel{\text{def}}{=} \{x_k : x_k \in W - V\}$ and
   Let $V' = V$.

5. **If** $\{Q = \emptyset\}$, **then** {goto 9}.

6. Choose $x_k \in Q$, set $V' = V \cup \{x_k\}$, and set $Q = Q - \{x_k\}$.

7. Goto 5 **if** for any $\{x_i, x_j\} \subseteq V'$, this condition is satisfied:
   $\exists\ x_l \in W - \{x_i, x_j\} : \hat{C}^l_{i,j} > \omega/2$
   OR
   $\exists\ \{x_l, x_m\} \subseteq W - \{x_i, x_j\} : \left|\hat{C}^m_{i,l} - \hat{C}^m_{j,l}\right| > \omega/2.$

8. Set $V = V'$ and goto 5.

9. **If** $\{|V| = 2\}$, **then** {set $W' = W' - V$, and goto 13}.

10. Choose $\{x_i, x_j\} \subset V$ and let $U = \{x_i, x_j\}$.

11. **If** $\{\exists\ x_k \in V - U :$
    $\left|\hat{\Gamma}_{p,q} - \hat{\Gamma}_{p,r}\right| > (1 - \gamma_n)/2$, AND $\left|\hat{\Gamma}_{p,q} - \hat{\Gamma}_{q,r}\right| > (1 - \gamma_n)/2$, AND
    $\left|\hat{\Gamma}_{p,r} - \hat{\Gamma}_{q,r}\right| \leq \gamma_n/2\}$, for $\{x_p, x_q, x_r\} = U \cup \{x_k\}$

then {set $x_i = x_p, x_j = x_q$, $x_k = x_r$, $V = V - \{x_k\}$, $U = \{x_i, x_j\}$, and goto 9};

12. **Else** {set $W' = W' - V$}.

13. **If** { $\hat{\Gamma}_{i,j} > 1/2$ for all $\{x_i, x_j\} \subseteq V$}, **then** {set $T = T \cup \{(V, \text{AND})\}$ and goto 1} **else** {set $T = T \cup \{(V, \text{OR})\}$ and goto 1}.

Note that the first 8 steps of the algorithm are for constructing a G-CMSLL candidate V. Step 9 considers this V as a G-CMSLL whenever it contains only two variables. Whenever the G-CMSLL candidate V contains three or more variables, if the condition in step 11 is satisfied, then the sequence of steps 9, 10, and 11 identifies a subset of V that can be considered as a G-CMSLL. Moreover, if the condition in step 11 is not satisfied, then step 12 considers the G-CMSLL candidate V as a G-CMSLL. For the G-CMSLL found in step 9 or 12, step 13 constructs a G-perceptron. Note that, in step 13, we have chosen an OR gate type for the G-perceptron whenever it can be either an AND or an OR gate.

3.9 Constructing the skeleton

Let us now describe how we are going to use our different algorithms to build the skeleton. First we need to run once **FindOR** and **FindAND** over the set of all relevant variables W. Recall that these algorithms will return the set $T_b = \{B_1, B_2, \cdots\}$ of all bottom level siblings of an OR gate and the set $T'_b = \{B'_1, B'_2, \cdots\}$ of all bottom level siblings of an AND gate. They will also return the sets $T_{nb} = \{(N_1, D_1), (N_2, D_2), \cdots\}$ and $T'_{nb} = \{(N'_1, D'_1), (N'_2, D'_2), \cdots\}$ where each N_i is a non-bottom level siblings to an OR or an AND gate and D_i is the set of all the descendents of the parent of N_i

(excluding N_i).

We now need to build the corresponding gate for each B_i. But recall that whenever we build a gate (or a perceptron) g, we need to remove from W and each D_i all the variables that feed g except one (denote it by x_i) which we call a group representative. If g happens to be the child of a gate g', then the D_i associated with g' will now contain, after the update, only one variable which is the group representative x_i of the newly formed gate. Hence, after building a perceptron (or gate) g, whenever we find a D_i that contains only a single variable x_i, we will need to build a gate g' whose children consist of N_i and the newly formed perceptron (or gate) g. This is an example of what we call a *skeleton fragment* and, whenever we build one, we will *bind* (*i.e.* associate) a group representative (here x_i) with that skeleton fragment. Previously, x_i was bound to g; it is now bound at the skeleton fragment rooted at g'. Finally, we will also need to remove this (N_i, D_i) from $T_{nb} \cup T'_{nb}$ and update W and the other D_js accordingly.

In fact, it is only when D_i contains a single variable that we will build a gate for that (N_i, D_i). This is justified by the fact that we can replace (without changing f) any gate g' whose children consist of a set N_i of variables plus two or more perceptrons/gates $g_1, g_2..g_k$ by the same gate g' having as children the same set N_i of variables plus only one gate g, of the same type as g', that has $g_1, g_2..g_k$ as children. If we now denote by x_i the group representative of the variables meeting at g_i, our previous lemmas show that **FindG-CMSLL** will identify the set $V = \{x_1, x_2..x_k\}$ as a G-CMSLL. Hence, the D_i associated with g' will contain only one variable as soon as the skeleton fragment associated with V will be formed.

Hence, whenever a CMSLL V has been identified (by **FindG/NG-CMSLL** or **FindOR/AND**) we will need a procedure **GrowFragment** that will choose a group

representative $x_i \in V$ and bind x_i with the gate type and the skeleton fragment rooted at that gate. Moreover, as explained above, **GrowFragment** will test if building such a fragment has left a D_i containing only x_i. If so, it will create a new gate (and thus a larger fragment) whose children consist of N_i and the previous skeleton fragment. It will continue growing this fragment in the upward direction until there does not exist any D_i containing only x_i.

Algorithm GrowFragment

Input: A CMSLL V and a gate type $t \in \{NG, OR, AND\}$ associated with V. The other inputs are the sets T_{nb} and T'_{nb} initially produced by **FindOR** and **FindAND** respectively and a set $W \supseteq V$ of relevant variables.

Output: A skeleton fragment and a variable bound to that fragment that we call the group representative of all the variables feeding that fragment. The set W is updated so as to contain no variable feeding the returned fragment except its group representative. T_{nb} and T'_{nb} are updated so to contain only pairs (N, D) for which D do not contain only the group representative.

1. Choose a group representative $x_i \in V$ and bind x_i with the skeleton fragment made of this perceptron of gate type t and all its descendents.
2. Set $W = (W - V) \cup \{x_i\}$.
3. Set $D = (D - V) \cup \{x_i\}$ for all $(N, D) \in (T_{nb} \cup T'_{nb}) : x_i \in D$.
4. **While** $(\exists\ (N, D) \in (T_{nb} \cup T'_{nb}) :\ D = \{x_i\})$, **do**
   {

- Build an OR gate for $(N, D) \in T_{nb}$ or an AND gate for $(N, D) \in T'_{nb}$ and bind x_i with the skeleton fragment made of this gate and all its descendents.
- Set $T_{nb} = T_{nb} - \{(N, D)\}$.
- Set $T'_{nb} = T'_{nb} - \{(N, D)\}$.
- Set $D' = D' - N$ for all $D' : x_i \in D'$.

}

Equipped with **growFragment**, the procedure for building the complete skeleton should now become clear. After running **FindOR** and **FindAND**, we run **GrowFragment** on each $B \in T_b \cup T'_b$ to grow several skeleton fragments consisting of only AND and OR gates. There is a variable (a group representative) associated to each of these skeleton fragments. The sets W, T_{nb}, and T'_{nb} will have been updated as described previously but we need, in addition, to remove all N_is from the remaining set W of relevant variables so that W constitutes a C-safe set on which we can run **FindNG/G-CMSLL**. Before running **FindG-CMSLL**, we will need to run repeatedly the sequence **FindNG-CMSLL** and **GrowFragment** up to the point where the remaining set W of relevant variables do not contain any NG-CMSLL of three or more variables. This algorithm, **BuildSkeleton**, will return the complete skeleton of the target function f when there is at most two relevant variables that remain in W.

Algorithm BuildSkeleton

Input: The set W of all relevant variables.

Output: The skeleton $\mathcal{S}$ of the target function f.

1. **FindOR**(W)

   It outputs $T_b = \{B_1, \cdots, B_r\}$ and $T_{nb} = \{(N_1, D_1), \cdots, (N_s, D_s)\}$.

2. **FindAND**(W)

   It outputs $T'_b = \{B'_1, \cdots, B'_p\}$ and $T'_{nb} = \{(N'_1, D'_1), \cdots, (N'_q, D'_q)\}$.

3. Set $W = W - [(\cup_{i=1}^{s} N_i) \cup (\cup_{i=1}^{q} N'_i)]$.

4. For all $B \in T_b$, do **GrowFragment**$(B, \text{OR}, W, T_{nb}, T'_{nb})$

5. For all $B' \in T'_b$, do **GrowFragment**$(B', \text{AND}, W, T_{nb}, T'_{nb})$

6. **While** $|W| \geq 3$, **do**

   {

   repeat

   {

   FindNG-CMSLL(W)

   It outputs $T = \{V_1, \cdots, V_k\}$.

   For all $V \in T$, do **GrowFragment**$(V, \text{NG}, W, T_{nb}, T'_{nb})$

   } **until** $T = \emptyset$.

   FindG-CMSLL(W)

   It outputs $T = \{(V_1, t_1) \cdots, (V_k, t_k)\}$.

   For all $(V, t) \in T$, do **GrowFragment**$(V, t, W, T_{nb}, T'_{nb})$

   }

7. **If** $|W| = 1$ contains only one variable x_i, **then** return the skeleton fragment associated to x_i as the complete skeleton.

8. **If** $|W| = 2$, **then** build a NG-perceptron for W and return the skeleton fragment made of this NG-perceptron and all its descendents as the complete skeleton.

After finding the skeleton, we need, in principle, to find a good threshold value for each NG-perceptron. However, we need to realize that, after removing some variables, the output of each NG-perceptron will generally be a stochastic function in terms of its remaining children. In these circumstances, we have chosen to model the stochastic behavior of each NG-perceptron with enough precision to ensure that the returned hypothesis will make a small error with high confidence.

3.10 Building stochastic NG-perceptrons

In this section, we introduce an additional statistical test that provides a good stochastic model for each of NG-perceptron from bottom up. We will then discuss that this leads to a PAC learning algorithm for NPN target functions.

Let us first consider a bottom level perceptron g (from now on, perceptrons refer only to NG-perceptrons of three or more variables) for which some of its children have been removed. We then have the following lemma.

Lemma 3.9 *Let g be a bottom level perceptron that has a set V of relevant children and a set U of irrelevant children. Let $\tilde{g}$ be the stochastic (or p-concept) perceptron obtained from g by ignoring all $x_i \in U$. Then, we have:*

$$\mathrm{err}(g, \tilde{g}) \leq 2 \sum_{i=1}^{|U|} \mathrm{Infl}_g(x_i) \Pr(x_i = 1) \Pr(x_i = 0)$$

Proof: From the definition of $\tilde{g}$, we have:

$$\Pr(\tilde{g} = 1 | S_V = l) = \Pr(g = 1 | S_V = l)$$

where $S_V \stackrel{\mathrm{def}}{=} \sum_{i \in V} x_i$. Hence, we have $\Pr(\tilde{g} = 1) = \Pr(g = 1)$ and $\mathrm{Infl}_{\tilde{g}}(x_i) = \mathrm{Infl}_g(x_i)$.

To prove the lemma, we are using the fact that the variation error is the same as the (usual) error $\Pr(g \neq \tilde{g})$ whenever at least one of the two functions involved is deterministic over the set of variables that is used to compute the variation error.

$$\begin{aligned}
\mathrm{err}(g,\tilde{g}) &= \sum_{l} \Pr(S_V = l) \sum_{a_1} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_1 = a_1) \cdots \Pr(x_U = a_U) \times \\
&\quad |\Pr(g = 1 | S_V = l, x_1 = a_1 \cdots x_U = a_U) - \Pr(\tilde{g} = 1 | S_V = l)| \\
&= \sum_{l} \Pr(S_V = l) \sum_{a_1} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_1 = a_1) \cdots \Pr(x_U = a_U) \times \\
&\quad |\Pr(g = 1 | S_V = l, x_1 = a_1 \cdots x_U = a_U) - \Pr(g = 1 | S_V = l)|
\end{aligned}$$

By adding and subtracting terms like $\Pr(g = 1 | S_V = l, x_i = a_i \cdots x_U = a_U)$ for $i = 2, \cdots, |U| - 1$ inside the absolute value, we find:

$$\begin{aligned}
\mathrm{err}(g,\tilde{g}) &= \sum_{l} \Pr(S_V = l) \sum_{a_1} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_1 = a_1) \cdots \Pr(x_U = a_U) \times \\
&\quad \Big| \sum_{i=1}^{|U|} \{ \Pr(g = 1 | S_V = l, x_i = a_i \cdots x_U = a_U) \\
&\qquad - \Pr(g = 1 | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U) \} \Big| \\
&\leq \sum_{l} \Pr(S_V = l) \sum_{a_1} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_1 = a_1) \cdots \Pr(x_U = a_U) \times \\
&\quad \sum_{i=1}^{|U|} \{ |\Pr(g = 1 | S_V = l, x_i = a_i \cdots x_U = a_U) \\
&\qquad - \Pr(g = 1 | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U)| \} \\
&= \sum_{l} \Pr(S_V = l) \sum_{i=1}^{|U|} \{ \\
&\quad \sum_{a_i} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_i = a_i) \cdots \Pr(x_U = a_U) \times \\
&\quad |\Pr(g = 1 | S_V = l, x_i = a_i \cdots x_U = a_U) \\
&\qquad - \Pr(g = 1 | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U)| \} \\
&= \sum_{l} \Pr(S_V = l) \sum_{i=1}^{|U|} \{
\end{aligned}$$

$$\sum_{a_{i+1}} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_{i+1} = a_{i+1}) \cdots \Pr(x_U = a_U) \times \{$$
$$\Pr(x_i = 1) | \Pr(g = 1 | S_V = l, x_i = 1, x_{i+1} = a_{i+1} \cdots x_U = a_U)$$
$$- \Pr(g = 1 | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U) |$$
$$+ \Pr(x_i = 0) | \Pr(g = 1 | S_V = l, x_i = 0, x_{i+1} = a_{i+1} \cdots x_U = a_U)$$
$$- \Pr(g = 1 | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U) | \} \}$$

$$= \sum_l \Pr(S_V = l) \sum_{i=1}^{|U|} \{$$
$$\sum_{a_{i+1}} \cdots \sum_{a_U \in \{0,1\}} \Pr(x_{i+1} = a_{i+1}) \cdots \Pr(x_U = a_U) \times \{$$
$$\Pr(x_i = 1) | \Pr(x_i = 0) \mathrm{Infl}_g(x_i | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U) | +$$
$$\Pr(x_i = 0) | \Pr(x_i = 1) \mathrm{Infl}_g(x_i | S_V = l, x_{i+1} = a_{i+1} \cdots x_U = a_U) | \} \}$$

$$= 2 \sum_{i=1}^{|U|} \mathrm{Infl}_g(x_i) \Pr(x_i = 1) \Pr(x_i = 0)$$

□

Unfortunately, we generally do not have enough information from the examples to find $\Pr(\tilde{g} = 1 | S_V = l)$ for all values of l. However, we can use an approximation $\tilde{g}'$ of $\tilde{g}$ which is good enough to satisfy the PAC criterion. We could define $\tilde{g}'$ to be the stochastic perceptron with the activation function given by:

$$\frac{\Pr(f = 1 | S_V = l) - \Pr(f = 1 | S_V = 0)}{\Pr(f = 1 | S_V = |V|) - \Pr(f = 1 | S_V = 0)} = \frac{\Pr(g = 1 | S_V = l) - \Pr(g = 1 | S_V = 0)}{\Pr(g = 1 | S_V = |V|) - \Pr(g = 1 | S_V = 0)}$$

However, since we have only access to a finite set of examples, let us instead define $\tilde{g}'$ by:

$$\Pr(\tilde{g}' = 1 | S_V = l) \stackrel{\text{def}}{=} \frac{\Pr(f = 1 | S_V = l) - \Pr(f = 1 | S_V = s^-)}{\Pr(f = 1 | S_V = s^+) - \Pr(f = 1 | S_V = s^-)} \tag{3.1}$$

where s^+ and s^- are, respectively, the maximum and minimum value for l for which we have $\Pr(S_V = l) > \epsilon_S$ for $s^- \leq l \leq s^+$.

Let $\alpha \stackrel{\text{def}}{=} \Pr(g = 1|S_V = s^-)$ and $\beta \stackrel{\text{def}}{=} \Pr(g = 0|S_V = s^+)$. Then:

$$\Pr(\tilde{g}' = 1|S_V = l) = \frac{\Pr(\tilde{g} = 1|S_V = l) - \alpha}{1 - (\alpha + \beta)}.$$

For $\alpha = \beta = 0$, we have $\Pr(\tilde{g}' = 1|S_V = l) = \Pr(\tilde{g} = 1|S_V = l)$ *i.e.* the desired one. For general values of α and β, we can express $\text{err}(g, \tilde{g}')$ in terms of $\text{err}(g, \tilde{g})$ and $\Pr(g = a)$:

$$\begin{aligned}
\text{err}(g, \tilde{g}') &= \Pr(g = 1, \tilde{g}' = 0) + \Pr(g = 0, \tilde{g}' = 1) \\
&= \sum_l \Pr(S_V = l)[\Pr(g = 1|S_V = l)\Pr(\tilde{g}' = 0|S_V = l) \\
&\qquad + \Pr(g = 0|S_V = l)\Pr(\tilde{g}' = 1|S_V = l)] \\
&= \Pr(g = 1) \\
&\quad + \sum_l \Pr(S_V = l)\Pr(\tilde{g}' = 1|S_V = l)[1 - 2\Pr(g = 1|S_V = l)] \\
&= \Pr(g = 1) \\
&\quad + \sum_l \Pr(S_V = l)\frac{\Pr(\tilde{g} = 1|S_V = l) - \alpha}{1 - (\alpha + \beta)}[1 - 2\Pr(g = 1|S_V = l)] \\
&= \Pr(g = 1) \\
&\quad + \sum_l \Pr(S_V = l)\frac{\Pr(\tilde{g} = 1|S_V = l)}{1 - (\alpha + \beta)}[1 - 2\Pr(g = 1|S_V = l)] \\
&\quad - \sum_l \Pr(S_V = l)\frac{\alpha}{1 - (\alpha + \beta)}[1 - 2\Pr(g = 1|S_V = l)] \\
&= \Pr(g = 1) + \frac{\text{err}(g, \tilde{g}) - \Pr(g = 1)}{1 - (\alpha + \beta)} - \frac{\alpha}{1 - (\alpha + \beta)} \\
&\quad + \frac{2\alpha}{1 - (\alpha + \beta)}\Pr(g = 1) \\
&= \frac{1}{1 - (\alpha + \beta)}\{\text{err}(g, \tilde{g}) - \beta\Pr(g = 1) - \alpha\Pr(g = 0)\}
\end{aligned}$$

When $\text{err}(g, \tilde{g}) \leq \Pr(g = 1)$ and $\text{err}(g, \tilde{g}) \leq \Pr(g = 0)$, we have $\text{err}(g, \tilde{g}') \leq \text{err}(g, \tilde{g})$. When $\text{err}(g, \tilde{g}) > \Pr(g = 1)$ or $\text{err}(g, \tilde{g}) > \Pr(g = 0)$, we may not have $\text{err}(g, \tilde{g}') \leq \text{err}(g, \tilde{g})$. However, we will show that replacing each perceptron g by the approximation $\tilde{g}'$ will give an hypothesis that does not make much error with the target function f.

The above argument can be applied to any non-bottom level perceptron. For this purpose, let us now consider a non-bottom level perceptron G having r children g_i for $i = 1, \cdots, r$. We then have the following lemma.

Lemma 3.10 *Let G be a non-bottom level perceptron. Let $V = \{g_1, g_2, \cdots, g_r\}$ be its children. Let $\tilde{G}$ be the stochastic (or p-concept) perceptron obtained from G by replacing each g_i by the approximation $\tilde{g}_i'$ defined earlier. Then we have:*

$$\mathrm{err}(G, \tilde{G}) \leq \sum_{i=1}^{r} \mathrm{Infl}_G(g_i)\mathrm{err}(g_i, \tilde{g}_i')$$

Proof: By the definition of $\tilde{G}$, we have:

$$\Pr(\tilde{G} = 1 | S_V = l) = \Pr(G = 1 | S_V = l)$$

where now $S_V \stackrel{\text{def}}{=} \sum_{i=1}^{r} \tilde{g}_i'$.

The set of all the variables that feeds G is represented by the vector $(\mathbf{x}_V, \mathbf{x}_U)$, where $\mathbf{x}_U$ represent the irrelevant variables and $\mathbf{x}_V$ the relevant (visible) variables. Since the output of G is deterministic over $(\mathbf{x}_V, \mathbf{x}_U)$, we can write:

$$\mathrm{err}(G, \tilde{G}) = \sum_{\mathbf{x}_V} \Pr(\mathbf{x}_V) \sum_{\mathbf{x}_U} \Pr(\mathbf{x}_U) \left| \Pr(G = 1 | \mathbf{x}_V, \mathbf{x}_U) - \Pr(\tilde{G} = 1 | \mathbf{x}_V) \right|$$

where the sums are over all possible values of $\mathbf{x}_V$ and $\mathbf{x}_U$. Note that G depends on $(\mathbf{x}_V, \mathbf{x}_U)$ while $\tilde{G}$ depends only on $\mathbf{x}_V$ through the $\tilde{g}_i'$s.

Let V_i be relevant children and U_i be irrelevant children of a perceptron g_i. Let $S_i \stackrel{\text{def}}{=} \sum_j x_j$ for $x_j \in V_i$ and $S_i' \stackrel{\text{def}}{=} \sum_j x_j$ for $x_j \in U_i$. Then, we have:

$$\begin{aligned} \mathrm{err}(G, \tilde{G}) = \; & \Pi_{i=1}^{r} \left(\sum_{l_i} \sum_{l_i'} \Pr(S_i = l_i) \Pr(S_i' = l_i') \right) \times \\ & \left| \Pr(G = 1 | S_1 = l_1, S_1' = l_1', \cdots, S_r = l_r, S_r' = l_r') \right. \\ & \left. - \Pr(\tilde{G} = 1 | S_1 = l_1, \cdots, S_r = l_r) \right| \end{aligned}$$

Let $\tilde{G}_i$ be obtained from $\tilde{G}_{i-1}$ by replacing g_i by g_i'. Then, by adding and subtracting terms like

$$\Pr(\tilde{G}_i = 1 | S_1 = l_1, \cdots, S_i = l_i, S_{i+1} = l_{i+1}, S'_{i+1} = l'_{i+1}, \cdots, S_r = l_r, S'_r = l'_r)$$

for $i = 2, \cdots r-1$ inside the absolute value and expanding over g_i and $\tilde{g}_i'$, we find:

$$\mathrm{err}(G, \tilde{G}) \leq \sum_{i=1}^{r} \mathrm{Infl}_{\tilde{G}_{i-1}}(g_i) \mathrm{err}(g_i, \tilde{g}_i')$$

By the definition of $\tilde{G}_i$, we have $\mathrm{Infl}_{\tilde{G}_{i-1}}(g_i) = \mathrm{Infl}_G(g_i)$. Therefore,

$$\mathrm{err}(G, \tilde{G}) \leq \sum_{i=1}^{r} \mathrm{Infl}_G(g_i) \mathrm{err}(g_i, \tilde{g}_i').$$

□

Like bottom level perceptrons, we are generally not able to find $\Pr(\tilde{G} = 1 | S_V = l)$ for all values of l; instead, we will use a similar approximation:

$$\begin{aligned} \Pr(\tilde{G}' = 1 | S_{V'} = l) &\stackrel{\text{def}}{=} \frac{\Pr(f = 1 | S_V = l) - \Pr(f = 1 | S_V = s^-)}{\Pr(f = 1 | S_V = s^+) - \Pr(f = 1 | S_V = s^-)} \\ &= \frac{\Pr(\tilde{G} = 1 | S_V = l) - \alpha}{1 - (\alpha + \beta)} \end{aligned} \tag{3.2}$$

where, here, we have $S_{V'} \stackrel{\text{def}}{=} \sum_i \tilde{g}_i'$, $\alpha \stackrel{\text{def}}{=} \Pr(G = 1 | S_V = s^-)$, and $\beta \stackrel{\text{def}}{=} \Pr(G = 0 | S_V = s^+)$.

Also, by using the same method as above, we can simply show that:

$$\mathrm{err}(G, \tilde{G}') = \frac{1}{1 - (\alpha + \beta)} \left\{ \mathrm{err}(G, \tilde{G}) - \beta \Pr(G = 1) - \alpha \Pr(G = 0) \right\}.$$

Similarly, when $\mathrm{err}(G, \tilde{G}) \leq \Pr(G = 1)$ and $\mathrm{err}(G, \tilde{G}) \leq \Pr(G = 0)$, we have: $\mathrm{err}(G, \tilde{G}') \leq \mathrm{err}(G, \tilde{G})$. But when $\mathrm{err}(G, \tilde{G}) > \Pr(G = 1)$ or $\mathrm{err}(G, \tilde{G}) > \Pr(G = 0)$, we may not have $\mathrm{err}(g, \tilde{g}') \leq \mathrm{err}(g, \tilde{g})$. But as we mentioned, replacing G by $\tilde{G}'$ will give an hypothesis that will not make much error with the target function.

Hence, to build the hypothesis from an NPN target function, we replace each perceptron by the above-defined approximation over its relevant children, from the bottom level perceptrons to the root. The next lemma ensures that the returned hypothesis will not make too much error with the target function.

Lemma 3.11 *Let f be an NPN target function. Let $\tilde{f}$ be a stochastic function obtained from f by replacing each perceptron g of the network by its approximation $\tilde{g}'$ as defined by eqs 3.1 and 3.2. Then:*

$$\mathrm{err}(f, \tilde{f}) \leq 2 \sum_{i=1}^{r} \mathrm{Infl}(x_i)\Pr(x_i = 1)\Pr(x_i = 0)$$

where r is the total number of variables that have been removed.

Proof: See appendix I . □

Lemma 3.11 tells us that such a $\tilde{f}$ makes a small error with f whenever the removed variables are either sticky or weakly influential.

The next algorithm formally describes the procedure used to approximate each NG-perceptron by a stochastic perceptron.

Algorithm Build-Stochastic-NG-Perceptrons

Input: The set W of all relevant variables and the skeleton $\mathcal{S}$ of the target function.

Output: The hypothesis function h containing a stochastic model of each NG-perceptron.

From the bottom-level perceptrons to the root, do the following steps for each NG-perceptron g in the skeleton $\mathcal{S}$:

1. Let $S \stackrel{\text{def}}{=} \sum_i x_i$ where the x_is are g's children Let s^- and s^+ be respectively the minimum and the maximum value for l for which we have $\hat{\Pr}(S = l) > \epsilon_S/2$.

2. **If** g is not the root node, **then** build the stochastic model $\tilde{g}'$ of g by computing for each $l \in \{s^-, s^- + 1 \cdots s^+\}$:

$$\Pr(\tilde{g}' = 1 | S_V = l) = \frac{\hat{\Pr}(f = 1 | S_V = l) - \hat{\Pr}(f = 1 | S_V = s^-)}{\hat{\Pr}(f = 1 | S_V = s^+) - \hat{\Pr}(f = 1 | S_V = s^-)};$$

else build the stochastic model of f by computing for each $l \in \{s^-, s^- + 1 \cdots s^+\}$: $\Pr(h = 1 | S_V = l) = \hat{\Pr}(f = 1 | S_V = l)$. Return this h has the hypothesis function.

3.11 The Learning Algorithm

Combining all the steps of the previous sections, we now have the following learning algorithm for learning μ-perceptron networks under product distributions.

Algorithm: LearnNPN (n, ϵ, δ)

Input: The set X of n input variables.

Output: The hypothesis function h

1. Call $m = \frac{1}{3}\left(\frac{6}{\epsilon_S}\right)^3 \left(\frac{8}{3\epsilon_I \gamma_n}\right)^6 \ln(\frac{12(4n)^3}{\delta})$ training examples.

2. Remove every x_i from X for which either $\hat{\Pr}(x_i = 1) < 3\epsilon/(4n)$ or $\hat{\Pr}(x_i = 1) > 1 - 3\epsilon/(4n)$. Remove every x_i from X for which $\left|\hat{\text{Infl}}(x_i)\right| < 3\epsilon/(16n)$. Let W be the remaining set of variables.

3. For every $x_i \in W$, let $w_i = +1$ if $\hat{\text{Infl}}(x_i) \geq 3\epsilon/(16n)$ and let $w_i = -1$ otherwise. Let $x_i = 1 - x_i$ whenever $w_i = -1$ (conversion into the positive form).

4. $\mathcal{S} =$ **BuildSkeleton**(W);

5. $h =$ **Build-Stochastic-NG-Perceptrons** $(W, \mathcal{S})$

6. Convert h from positive to normal form and return the resulting h.

Finally, we have the following theorem:

Theorem 3.1 *Algorithm* **LearnNPN** *PAC learns the class of nonoverlapping perceptrons under product distributions. The number of examples needed is:*

$$m = (1/3)\,(6/\epsilon_S)^3\,(8/(3\epsilon_I\gamma_n))^6 \ln(12(4n)^3/\delta)$$

for the returned hypothesis to have $\text{err}(h, f) < \epsilon$ *with probability* $> 1 - \delta$. *The running time of* **LearnNPN** *is of* $O(m \times n^3)$.

Proof: If u_S sticky variables and u_I weakly influential variables have been removed from the target function f, the previous lemmas indicate that the returned hypothesis h will make an error of at most $u_S\epsilon_S + 2u_I\epsilon_I$ with a confidence of at least $1-\delta$ whenever m is sufficiently large to have simultaneously:

- $|\hat{\Pr}(x_i = b) - \Pr((x_i = b)| < \epsilon_S/2$ for $b \in \{0, 1\}$ with confidence at least $1-\delta/(4n)$
- $|\hat{\text{Infl}}(x_i) - \text{Infl}(x_i)| < \epsilon_I/2$ with confidence at least $1 - \delta/(4n)$
- $|\hat{A^b_{j,i}} - A^b_{j,i}| < \gamma_n/2$ with confidence at least $1 - \delta/(16n^2)$
- $|\hat{C^k_{i,j}} - C^k_{i,j}| < \epsilon_I\gamma_n^2/4$ with confidence at least $1 - \delta/(8n^3)$

- $|\hat{\Gamma_{i,j}} - \Gamma_{i,j}| < \gamma_n/2$ with confidence at least $1 - \delta/(16n^2)$.

If we choose m in order to satisfy the fourth inequality with the required confidence, all the inequalities above will be satisfied (with the required confidence). That value of m can be found by using Chernoff bounds [11] (see appendix J for details). By choosing $\epsilon_I = \epsilon_f/(8n)$, $\epsilon_S = \epsilon_f/(2n)$, and $\epsilon_f = \epsilon$ the error will be at most ϵ with confidence at least $1 - \delta$. □

Chapter 4

Conclusion

In this thesis, we have investigated different ways to overcome the difficulties of learning in some restricted neural networks.

In chapter 2, we came up with a PAC learning algorithm for a single stochastic perceptron under k-blocking distributions. Although this class of distributions is quite large, the learner had to have some prior knowledge about the distribution: the blocking set of each variable. Of course, we would like to have an algorithm that does not need this prior knowledge but the hardness result of Hoeffgen [14] implies that this is not likely to happen.

In chapter 3, we presented a statistical method that PAC learns the class of μ-perceptron networks with binary weights under product distributions. The next logical step would be to try to extend the learning algorithm either to a larger class of distributions or a larger class of functions.

On the distribution side, there is hope that this method extends to k-blocking distributions. But even the restricted case of product distributions can find applications in practice. Indeed, many practitioners often perform a Principal Component Analysis

on the training set to obtain new variables which are only weakly correlated. Since the resulting components are generally real valued, we would then need to extend our methods from the Boolean to the Real domain. We have good hope that this is indeed possible.

On the function side, we have good hope that our methods could be extended to nonoverlapping networks of stochastic perceptrons. Indeed, by removing weakly relevant variables, the target function became stochastic. Hence, we would need to see if it would be possible to adapt our methods to the case where the stochastic behavior does not come from hidden variables. Also, it would be interesting to extend to the case of real valued weights but this seems to be a very hard problem for the network case.

Finally there is always this relevant criticism about much of the research in Computational Learning Theory: how can we use in practice these theoretical PAC-learning algorithms? Indeed, the number of examples needed by our algorithms, although polynomial, is too large to be of any use in practice. Moreover, we *did* make a substantial effort to reduce our bounds but the ones presented in this thesis are the smallest we have been able to find. Hence, perhaps we simply cannot reduce them (significantly) further. Does anyone have (provably) better learning algorithms for these function classes?

Appendix A

Estimation of $m_0(\epsilon, \delta, n)$ in Chapter 2

Here, we present an estimation of the number m_0 of required examples discussed in Chapter 2 to have $\left|\hat{\text{Binf}}(x_i|\mathbf{b}_i) - \text{Binf}(x_i|\mathbf{b}_i)\right| < \epsilon$ with probability at least $1 - \delta$.

From the definition of $\text{Binf}(x_i|\mathbf{b}_i)$ in terms of probabilities, we can see that we need to have:

$$\left|\hat{\Pr}(\sigma = 1|\mathbf{b}_i, x_i) - \Pr(\sigma = 1|\mathbf{b}_i, x_i)\right| < \epsilon/2 \tag{A.1}$$

with probability at least $1 - \delta/2$.

Now let $|\hat{\Pr}(\sigma = 1 \wedge \mathbf{b}_i, x_i) - \Pr(\sigma = 1 \wedge \mathbf{b}_i, x_i)| < \epsilon'$ and $|\hat{\Pr}(\mathbf{b}_i, x_i) - \Pr(\mathbf{b}_i, x_i)| < \epsilon'$ each with probability at least $1 - \delta'$. Then, by using the same method as Pagallo and Haussler [24], we have:

$$\begin{aligned}
\hat{\Pr}(\sigma = 1|\mathbf{b}_i, x_i) &= \frac{\hat{\Pr}(\sigma = 1 \wedge \mathbf{b}_i, x_i)}{\hat{\Pr}(\mathbf{b}_i, x_i)} \\
&< \frac{\Pr(\sigma = 1 \wedge \mathbf{b}_i, x_i) + \epsilon'}{\Pr(\mathbf{b}_i, x_i) - \epsilon'} \\
&= \left(\Pr(\sigma = 1|\mathbf{b}_i, x_i) + \frac{\epsilon'}{\Pr(\mathbf{b}_i, x_i)}\right)\left(1 - \frac{\epsilon'}{\Pr(\mathbf{b}_i, x_i)}\right)^{-1} \\
&\leq \left(\Pr(\sigma = 1|\mathbf{b}_i, x_i) + \frac{\epsilon'}{\Pr(\mathbf{b}_i, x_i)}\right)\left(1 + 2\frac{\epsilon'}{\Pr(\mathbf{b}_i, x_i)}\right)
\end{aligned}$$

$$\leq \Pr(\sigma = 1|\mathbf{b}_i, x_i) + \frac{4\epsilon'}{\Pr(\mathbf{b}_i, x_i)} \tag{A.2}$$

where we have used the fact that for $z = \frac{\epsilon'}{\Pr(\mathbf{b}_i, x_i)}$ we have both $(1-z)^{-1} \leq 1 + 2z$ and $2z^2 \leq z$ whenever $z \leq 1/2$. Hence, it follows that we need to have:

$$\epsilon' \leq \Pr(\mathbf{b}_i, x_i)/2. \tag{A.3}$$

Similarly, we also need to have:

$$\hat{\Pr}(\sigma = 1|\mathbf{b}_i, x_i) > \Pr(\sigma = 1|\mathbf{b}_i, x_i) - \frac{4\epsilon'}{\Pr(\mathbf{b}_i, x_i)}.$$

That is:

$$\left|\hat{\Pr}(\sigma = 1|\mathbf{b}_i, x_i) - \Pr(\sigma = 1|\mathbf{b}_i, x_i)\right| < \frac{4\epsilon'}{\Pr(\mathbf{b}_i, x_i)}$$

provided that $\epsilon' \leq \Pr(\mathbf{b}_i, x_i)/2$.

Both conditions in eqs. A.1 and A.3 will be satisfied if we set:

$$\frac{4\epsilon'}{\Pr(\mathbf{b}_i, x_i)} < \frac{\epsilon}{2}$$

that is:

$$\epsilon' < \frac{\epsilon \Pr(\mathbf{b}_i, x_i)}{8}.$$

From lemma 2.1, since the lowest permissible value for $\Pr(\mathbf{b}_i, x_i)$ is $\kappa = \alpha^{k+1}$, then we set:

$$\epsilon' < \frac{\epsilon\kappa}{8}.$$

On the other hand, since we want $1 - \delta/2$ confidence for $|\hat{\Pr}(\sigma = 1|\mathbf{b}_i, x_i) - \Pr(\sigma = 1|\mathbf{b}_i, x_i)| < \epsilon/2$, we need to have $\delta' = \delta/4$. By using lemma 2.3 and the bound on ϵ', we now have:

$$\Pr\left(|\hat{\Pr}(\mathbf{b}_i, x_i) - \Pr(\mathbf{b}_i, x_i)| > \epsilon'\right) \leq 2\exp\left(-2m_0\epsilon'^2\right) \leq \delta/4$$

Hence, the value of m_0 needed to satisfy the last inequality is:

$$m_0(\epsilon, \delta, n) \geq \frac{1}{2}\left(\frac{8}{\kappa\epsilon}\right)^2 \ln\left(\frac{8}{\delta}\right).$$

Appendix B

Sample Complexity of Algorithm LearnSP (Chapter 2)

Here, we present the details on how to find the number of examples, m, needed by **LearnSP**, to have $err(h, c) < \epsilon$ with probability at least $1 - \delta$.

Let $c_{V'}$ be obtained from c by deleting the variables $x_i \in V_1$ that have $\Pr(x_i = a_i) < \alpha$ (lemma 2.1) and $x_i \in V_2$ that have $|\text{Binf}(x_i)| < \alpha'$(lemma 2.4), then:

$$err(c_{V'}, c) \leq |V_1|\alpha + |V_2|\alpha'$$

and without loss of generality, we can assume $\alpha' = \alpha$ and $V_1 + V_2 = V$. Hence, we can write:

$$err(c_{V'}, c) \leq |V|\alpha.$$

On the other hand, by the triangle inequality, we have:

$$\begin{aligned} err(h, c) &\leq err(h, c_{V'}) + err(c_V, c) \\ &\leq err(h, c_V) + |V|\alpha \end{aligned}$$

where the same variables remain in h and c_V. We can now explicitly write:

$$err(h, c_V) = \sum_s \Pr(\sum_i w_i x_i = s) \times \left| \hat{\Pr}(\sigma = 1 | \sum_i w_i x_i = s) - \Pr(\sigma = 1 | \sum_i w_i x_i = s) \right|.$$

Let us now divide the set of s values in two disjoint sets S_1 and S_2 such that $\Pr(\sum_i w_i x_i = s) \stackrel{\text{def}}{=} \Pr(s) \leq \gamma_1$ for $s \in S_1$ and $\Pr(s) > \gamma_1$ for $s \in S_2$. Let also $\left|\hat{\Pr}(\sigma = 1|s) - \Pr(\sigma = 1|s)\right| < \gamma_2$ for $s \in S_2$. Then:

$$\begin{aligned} err(h, c) &\leq \sum_{s \in S_1} \Pr(s) |\hat{\Pr}(\sigma = 1|s) - \Pr(\sigma = 1|s)| + \\ &\quad \sum_{s \in S_2} \Pr(s) |\hat{\Pr}(\sigma = 1|s) - \Pr(\sigma = 1|s)| + |V|\alpha \\ &\leq |S_1|\gamma_1 + \gamma_2 + |V|\alpha \\ &\leq (n+1)\gamma_1 + \gamma_2 + n\alpha. \end{aligned}$$

We now want $err(h, c)$ to be less than ϵ. Therefore, it is sufficient to choose:

$$\gamma_1 = \frac{\epsilon}{4(n+1)} \quad , \quad \gamma_2 = \frac{\epsilon}{4} \quad , \quad \alpha = \frac{\epsilon}{2n}.$$

Moreover, since we want to have at least $1 - \delta$ confidence for $err(h, c) < \epsilon$, we can choose a reasonable confidence for each separated region (bounded by $\alpha, \gamma_1,$ and γ_2).

Now, from lemma 2.1, the variables that remain in the hypothesis must have either $\Pr(x_i = +1) > \alpha$ or $\Pr(x_i = -1) > \alpha$. To have this, it is sufficient to have:

$$\hat{\Pr}(x_i = +1) > \frac{3\alpha}{2}$$

and

$$\left|\hat{\Pr}(x_i = +1) - \Pr(x_i = +1)\right| < \frac{\alpha}{2}.$$

By using the value of α, we find:

$$\hat{\Pr}(x_i = +1) > \frac{3\epsilon}{4n} \quad \text{(step 2 in algorithm)}$$

and

$$\left|\hat{\Pr}(x_i = +1) - \Pr(x_i = +1)\right| < \frac{\epsilon}{4n} \quad \text{with confidence at least } 1 - \frac{\delta}{4n}$$

where here we have chosen $1-\delta/(4n)$ confidence which reduces to $1-\delta/4$ for n variables. Then, from lemma 2.3, it can easily be seen that the sample size must be:

$$m_1 \geq \frac{1}{2}\left(\frac{4n}{\epsilon}\right)^2 \ln\left(\frac{8n}{\delta}\right).$$

From lemma 2.4, we need to have $|\text{Binf}(x_i)| > \alpha$. To have this, it is also sufficient to have:

$$\left|\hat{\text{Binf}}(x_i)\right| > \frac{3\alpha}{2}$$

and

$$\left|\hat{\text{Binf}}(x_i) - \text{Binf}(x_i)\right| < \frac{\alpha}{2}$$

Again by using the value of α, we find:

$$\left|\hat{\text{Binf}}(x_i)\right| > \frac{3\epsilon}{4n} \quad \text{(step 4 in algorithm)}$$

and

$$\left|\hat{\text{Binf}}(x_i) - \text{Binf}(x_i)\right| < \frac{\epsilon}{4n} \quad \text{with confidence at least } 1 - \frac{\delta}{4n}$$

where we have chosen again $1 - \delta/(4n)$ confidence which reduces to $1 - \delta/4$ for n variables. From appendix A, this is achieved whenever the sample size satisfies:

$$m_2 \geq 128\left(\frac{2n}{\epsilon}\right)^{2k+4} \ln\left(\frac{32n}{\delta}\right).$$

Since we must also have $|\hat{\Pr}(\sigma = 1|s) - \Pr(\sigma = 1|s)| < \gamma_2 = \epsilon/4$ (with confidence at least $1 - \delta/4$) and $\Pr(s) > \gamma_1$, from appendix A, we need to have:

$$\left|\hat{\Pr}(\sum_{i=1}^{n} w_i x_i = s) - \Pr(\sum_{i=1}^{n} w_i x_i = s)\right| < \left(\frac{\gamma_1\gamma_2}{4}\right)$$
$$= \frac{\epsilon^2}{64(n+1)} < \frac{\epsilon}{64(n+1)}$$

(with confidence at least $1 - \delta/[4(n+1)]$ which reduces to $1 - \delta/4$ for $n+1$ possible values for s). From lemma 2.3 again, the number of examples needed to satisfy this equation is:

$$m_3 \geq \frac{1}{2}\left[\frac{64(n+1)}{\epsilon}\right]^2 \ln\left(\frac{8(n+1)}{\delta}\right).$$

Also to have $\Pr(\sum_{i=1}^{n} w_i x_i = s) > \epsilon/[4(n+1)]$, we must have:

$$\hat{\Pr}(\sum_{i=1}^{n} w_i x_i = s) > \frac{\epsilon}{4(n+1)} + \frac{\epsilon}{64(n+1)} = \frac{17\epsilon}{64(n+1)}$$

(step 6 in algorithm).

Thus, by choosing the sample complexity $m = m_2$ (the largest one), we have an algorithm (**LearnSP**) that returns a hypothesis which makes an error at most ϵ (with the target function) with confidence at least $1 - \delta$.

Appendix C

Proof of Lemma 3.2 in Chapter 3

Here, we present the proof of Lemma 3.2 in Chapter 3. If there is a constant value $a \in \{0,1\}$ such that: $\mathrm{err}(f,a) = \Pr(f \neq a) < \epsilon_f$, then f can be replaced by a without making error more than ϵ_f. Hence, we let $\epsilon_f < \Pr(f = a) < 1 - \epsilon_f$ for any $a \in \{0,1\}$.

Let f_i be the function obtained from f_{i-1} by replacing the reading of x_i by the constant value a_i. Let $f_0 \stackrel{\text{def}}{=} f$ and $f_u \stackrel{\text{def}}{=} f_U$. Using the triangle inequality, we have:

$$\mathrm{err}(f_i, f) \;\leq\; \mathrm{err}(f_{i-1}, f) + \mathrm{err}(f_i, f_{i-1}) \text{ for } i = 1, 2, \cdots, u \;.$$

To prove lemma 3.2, we will first prove the two following claims:

$$\mathrm{err}(f_i, f_{i-1}) \;=\; \Pr(x_i \neq a_i)\left|\mathrm{Infl}_{f_{i-1}}(x_i)\right| \quad \text{(claim 1)}$$

$$\left|\mathrm{Infl}_{f_{i-1}}(x_i)\right| \;\leq\; \frac{|\mathrm{Infl}_f(x_i)|}{\mathrm{Infl}_f(f_{i-1})} \quad \text{(claim 2)}$$

The proof of the first claim is the same for each value of i. To shorten the notation we set, w.l.o.g., $i = 1$ and use the following definitions:

$$f_1^1(\mathbf{x}) \stackrel{\text{def}}{=} f_0(\mathbf{x}|x_1 = 1)$$

$$f_1^0(\mathbf{x}) \stackrel{\text{def}}{=} f_0(\mathbf{x}|x_1 = 0)$$

Recall that f_1 is obtained from f_0 by replacing x_1 by a_1. Then,

$$\begin{aligned}
\text{err}(f_1, f_0) &= \text{err}(f_1, f_0|x_1 = a_1)\text{Pr}(x_1 = a_1) \\
&\quad + \text{err}(f_1, f_0|x_1 \neq a_1)\text{Pr}(x_1 \neq a_1) \\
&= 0 + \text{err}(f_1, f_0|x_1 \neq a_1)\text{Pr}(x_1 \neq a_1) \\
&= \text{err}(f_1^1, f_1^0)\text{Pr}(x_1 \neq a_1)
\end{aligned}$$

and

$$\text{err}(f_1^1, f_1^0) = \text{Pr}(f_1^1 = 1|f_1^0 = 0)\text{Pr}(f_1^0 = 0) + \text{Pr}(f_1^1 = 0|f_1^0 = 1)\text{Pr}(f_1^0 = 1).$$

Since these functions are monotonic, we have:

$$\text{Pr}(f_1^1 = 0|f_1^0 = 1) = 0 \text{ if } w_1 = +1$$

and

$$\text{Pr}(f_1^1 = 1|f_1^0 = 0) = 0 \text{ if } w_1 = -1 \ .$$

Let $w_1 = +1$. Then,

$$\begin{aligned}
\text{err}(f_1^1, f_1^0) &= \text{Pr}(f_1^1 = 1|f_1^0 = 0)\text{Pr}(f_1^0 = 0) + 0 \\
&= \text{Pr}(f_1^0 = 0|f_1^1 = 1)\text{Pr}(f_1^1 = 1) \\
&= \text{Pr}(f_1^1 = 1) - \text{Pr}(f_1^0 = 1|f_1^1 = 1)\text{Pr}(f_1^1 = 1) \\
&= \text{Pr}(f_1^1 = 1) - \text{Pr}(f_1^1 = 1|f_1^0 = 1)\text{Pr}(f_1^0 = 1) \\
&= \text{Pr}(f_1^1 = 1) - \text{Pr}(f_1^0 = 1) \\
&= \text{Pr}(f_0 = 1|x_1 = 1) - \text{Pr}(f_0 = 1|x_1 = 0) \\
&= \text{Infl}_{f_0}(x_1) \ .
\end{aligned}$$

Let $w_1 = -1$. Then,

$$\begin{aligned}
\mathrm{err}(f_1^1, f_1^0) &= 0 + \Pr(f_1^1 = 0 | f_1^0 = 1)\Pr(f_1^0 = 1) \\
&= \Pr(f_1^0 = 1 | f_1^1 = 0)\Pr(f_1^1 = 0) \\
&= \Pr(f_1^1 = 0) - \Pr(f_1^0 = 0 | f_1^1 = 0)\Pr(f_1^1 = 0) \\
&= \Pr(f_1^1 = 0) - \Pr(f_1^1 = 0 | f_1^0 = 0)\Pr(f_1^0 = 0) \\
&= \Pr(f_1^1 = 0) - \Pr(f_1^0 = 0) \\
&= \Pr(f_0 = 0 | x_1 = 1) - \Pr(f_0 = 0 | x_1 = 0) \\
&= -\mathrm{Infl}_{f_0}(x_1) \; .
\end{aligned}$$

So,

$$\mathrm{err}(f_1, f_0) = \Pr(x_1 \neq a_1) \, |\mathrm{Infl}_{f_0}(x_1)|$$

and in general:

$$\mathrm{err}(f_i, f_{i-1}) = \Pr(x_i \neq a_i) \, \left|\mathrm{Infl}_{f_{i-1}}(x_i)\right| \; .$$

This proves claim 1.

To prove claim 2. we exploit again the fact that f is monotonic. Then:

$$w_i \mathrm{Infl}_f(x_i) = w_i \left[\Pr(f = 1 | x_i = 1) - \Pr(f = 1 | x_i = 0)\right] > 0 \; .$$

For any monotonic function f_{i-1}:

$$\begin{aligned}
w_i \mathrm{Infl}_f(x_i) &= w_i [\Pr(f = 1 | f_{i-1} = 1, x_i = 1)\Pr(f_{i-1} = 1 | x_i = 1) \\
&\quad + \Pr(f = 1 | f_{i-1} = 0, x_i = 1)\Pr(f_{i-1} = 0 | x_i = 1) \\
&\quad - \Pr(f = 1 | f_{i-1} = 1, x_i = 0)\Pr(f_{i-1} = 1 | x_i = 0) \\
&\quad - \Pr(f = 1 | f_{i-1} = 0, x_i = 0)\Pr(f_{i-1} = 0 | x_i = 0)]
\end{aligned}$$

Since $\Pr(f = 1|f_{i-1} = 1, x_i = 1) \geq \Pr(f = 1|f_{i-1} = 1, x_i = 0)$ and $\Pr(f = 1|f_{i-1} = 0, x_i = 1) \geq \Pr(f = 1|f_{i-1} = 0, x_i = 0)$, we then have:

$$w_i \mathrm{Infl}_f(x_i) \geq w_i \mathrm{Infl}_f(f_{i-1}|x_i = 1)\mathrm{Infl}_{f_{i-1}}(x_i)$$

and

$$w_i \mathrm{Infl}_f(x_i) \geq w_i \mathrm{Infl}_f(f_{i-1}|x_i = 0)\mathrm{Infl}_{f_{i-1}}(x_i).$$

By multiplying these two equations by $\Pr(x_i = 1)$ and $\Pr(x_i = 0)$ respectively, and adding them together, we **establish claim 2**.

Returning to the triangle inequality, we then have:

$$\begin{aligned} \mathrm{err}(f_i, f) &\leq \mathrm{err}(f_{i-1}, f) + \Pr(x_i \neq a_i) \left|\mathrm{Infl}_{f_{i-1}}(x_i)\right| \\ &\leq \mathrm{err}(f_{i-1}, f) + \Pr(x_i \neq a_i) \frac{|\mathrm{Infl}(x_i)|}{\mathrm{Infl}(f_{i-1})} \end{aligned}$$

We also have:

$$\begin{aligned} \mathrm{Infl}(f_{i-1}) &= \Pr(f = 1|f_{i-1} = 1) - \Pr(f = 1|f_{i-1} = 0) \\ &= 1 - [\Pr(f = 0|f_{i-1} = 1) + \Pr(f = 1|f_{i-1} = 0)] \\ &\geq 1 - \frac{\mathrm{err}(f, f_{i-1})}{\min[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)]}. \end{aligned}$$

Let us assume for now that:

$$\min[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)] \geq \frac{\epsilon_f}{2} \text{ for } i = 1, 2, \cdots, u.$$

We will show later that, otherwise, we have: $\Pr(f = a) < \epsilon_f$ for some $a \in \{0, 1\}$ which is not true by the hypothesis.

So if we assume that $\min[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)] \geq \epsilon_f/2$ for $i = 1, 2, \cdots, u$, we have:

$$\mathrm{Infl}(f_{i-1}) \geq 1 - \frac{2}{\epsilon_f}\mathrm{err}(f, f_{i-1}).$$

Therefore,

$$\mathrm{err}(f_i, f) \leq \mathrm{err}(f_{i-1}, f) + \Pr(x_i \neq a_i)\frac{|\mathrm{Infl}(x_i)|}{1 - \frac{2}{\epsilon_f}\mathrm{err}(f, f_{i-1})} \qquad \text{(C.1)}$$

This inequality along with the fact that $\mathrm{err}(f_1, f) = \Pr(x_1 \neq a_1)\,|\mathrm{Infl}(x_1)| < 2\epsilon_I$ hints us to complete the proof by induction. So, let the lemma hold for $i = k$ *i.e.* $\mathrm{err}(f_k, f) < 2k\epsilon_I$. Then, for the lemma to hold for $i = k+1$, we need that:

$$\mathrm{err}(f_{k+1}, f_k) < 2\epsilon_I$$

But since we have:

$$\begin{aligned} \mathrm{err}(f_{k+1}, f_k) &\leq \Pr(x_{k+1} \neq a_{k+1})\frac{|\mathrm{Infl}(x_{k+1})|}{1 - \frac{2}{\epsilon_f}\mathrm{err}(f, f_k)} \\ &< \Pr(x_{k+1} \neq a_{k+1})\frac{\epsilon_I}{1 - \frac{2}{\epsilon_f}(2k\epsilon_I)} \end{aligned}$$

the desired result is obtained whenever $\epsilon_I \leq \epsilon_f/(8n)$.

To finalize the proof we now show that if $\min[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)] < \epsilon_f/2$ for some $i \in \{1, 2, \cdots, u\}$, then we must have $\Pr(f = a) < \epsilon_f$ for some $a \in \{0, 1\}$.

To do so, let $l \in \{1, 2, \cdots, u\}$ be the first number for which we have $\min[\Pr(f_l = 1), \Pr(f_l = 0)] < \epsilon_f/2$. This means that for any $i \in \{1, 2, \cdots, l\}$, we have:

$$\min[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)] \geq \frac{\epsilon_f}{2}\ .$$

Therefore, equation C.1 holds for $i = 1 \cdots l$. Hence, we have $\mathrm{err}(f, f_l) < 2l\epsilon_I$.

Now let $a \in \{0, 1\}$ and let $\overline{a}$ be the negation of a. Then, we find:

$$\begin{aligned} \Pr(f = a) &= \mathrm{err}(f, \overline{a}) \\ &\leq \mathrm{err}(f, f_l) + \mathrm{err}(f_l, \overline{a}) \end{aligned}$$

$$\begin{aligned} &= \mathrm{err}(f, f_l) + \Pr(f_l = a) \\ &< 2l\epsilon_f + \frac{\epsilon_f}{2} \\ &\leq \epsilon_f \end{aligned}$$

while, by hypothesis, $\Pr(f = a) > \epsilon_f$ for $a \in \{0, 1\}$. This contradiction shows that we must have:

$$\min\left[\Pr(f_{i-1} = 1), \Pr(f_{i-1} = 0)\right] \geq \frac{\epsilon_f}{2} \text{ for } i = 1, 2, \cdots, u$$

as we assumed.

Appendix D

Proof of Lemma 3.3 in Chapter 3

Here, we present the proof of lemma 3.3 in chapter 3.

Let $\{x_1, x_2 \cdots, x_v\}$ be v independent random Boolean variables, each with $\Pr(x_i = 1) = q_i$ and let $S \stackrel{\text{def}}{=} \sum_{i=1}^{v} x_i$. Then, we first want to prove that:

$$\frac{1}{k} \cdot \frac{\Pr(S = k-1)}{\Pr(S = k)} \geq \frac{1}{k-1} \cdot \frac{\Pr(S = k-2)}{\Pr(S = k-1)} \tag{D.1}$$

for any $\{q_1, \cdots, q_v\}$ and any $k \in \{2, \cdots, v\}$.

This can easily be proven by induction on v. To do so, we first prove it for $v = 2$ and $k = 2$.

Let $S_2 \stackrel{\text{def}}{=} x_1 + x_2$. To prove for $v = 2$ and $k = 2$, it is sufficient to show that:

$$[\Pr(S_2 = 1)]^2 \geq 2 \cdot \Pr(S_2 = 0)\Pr(S_2 = 2).$$

This is indeed the case since:

$$\begin{aligned} \Pr(S_2 = 1)^2 &= [q_1(1-q_2) + (1-q_1)q_2]^2 \\ &= q_1^2(1-q_2)^2 + (1-q_1)^2 q_2^2 + 2q_1q_2(1-q_2)(1-q_1) \end{aligned}$$

and

$$2 \cdot \Pr(S_2 = 0)\Pr(S_2 = 2) = 2(1 - q_2)(1 - q_1)q_1q_2$$

where $q_1 = \Pr(x_1 = 1)$ and $q_2 = \Pr(x_2 = 1)$.

We now assume that inequality D.1 holds for v and any $k \in \{2, \cdots, v\}$, we now prove it for $(v + 1)$ and any $k \in \{2, \cdots, v + 1\}$. To do so, let $S' \stackrel{\text{def}}{=} S + x_{v+1}$ where $S = \sum_{i=1}^{v} x_i$. Hence, we need to show that:

$$\frac{1}{k} \cdot \frac{\Pr(S' = k - 1)}{\Pr(S' = k)} \geq \frac{1}{k - 1} \cdot \frac{\Pr(S' = k - 2)}{\Pr(S' = k - 1)} \tag{D.2}$$

for any $k \in \{2, \cdots, v + 1\}$, inequality D.2 holds if we show that:

$$\Pr(S' = k - 1)^2 \geq \frac{k}{k - 1} \cdot \Pr(S' = k - 2)\Pr(S' = k) \tag{D.3}$$

for any $k \in \{2, \cdots, v + 1\}$.

Note that we can write:

$$\begin{aligned} \Pr(S' = k - 1)^2 &= (1 - q_{v+1})^2\Pr(S = k - 1)^2 + q_{v+1}^2\Pr(S = k - 2)^2 \\ &\quad + 2q_{v+1}(1 - q_{v+1})\Pr(S = k - 1)\Pr(S = k - 2) \end{aligned} \tag{D.4}$$

and

$$\begin{aligned} \Pr(S' = k)\Pr(S' = k - 2) &= (1 - q_{v+1})^2\Pr(S = k)\Pr(S = k - 2) \\ &\quad + q_{v+1}^2\Pr(S = k - 1)\Pr(S = k - 3) \\ &\quad + q_{v+1}(1 - q_{v+1}) \times [\Pr(S = k - 1)\Pr(S = k - 2) \\ &\quad + \Pr(S = k)\Pr(S = k - 3)]. \end{aligned}$$

By using the fact that inequality D.1 holds for v and any $k \in \{2, \cdots, v\}$, we have:

$$\Pr(S = k - 1)^2 \geq \frac{k}{k - 1} \cdot \Pr(S = k - 2)\Pr(S = k)$$

and

$$\begin{aligned}\Pr(S=k-2)^2 &\geq \frac{k-1}{k-2}\cdot\Pr(S=k-1)\Pr(S'=k-3)\\ &> \frac{k}{k-1}\cdot\Pr(S=k-1)\Pr(S'=k-3)\end{aligned}$$

for any $k\in\{2,\cdots,v\}$.

Therefore, inequality D.3 holds for $(v+1)$ and any $k\in\{2,\cdots,v\}$ if we have:

$$\begin{aligned}2\Pr(S=k-1)\Pr(S=k-2) \geq\ & \frac{k}{k-1}\cdot[\Pr(S=k-1)\Pr(S=k-2)\\ &+\Pr(S=k)\Pr(S=k-3)]\end{aligned}$$

or if we have:

$$[2(k-1)-k]\Pr(S=k-1)\Pr(S=k-2)\geq k\Pr(S=k)\Pr(S=k-3)$$

or

$$\frac{1}{k}\frac{\Pr(S=k-1)}{\Pr(S=k)}\geq\frac{1}{k-2}\frac{\Pr(S=k-3)}{\Pr(S=k-2)}.$$

But, the above inequality is satisfied by our assumption, since we have:

$$\frac{1}{k}\frac{\Pr(S=k-1)}{\Pr(S=k)}\geq\frac{1}{k-1}\frac{\Pr(S=k-2)}{\Pr(S=k-1)}\geq\frac{1}{k-2}\frac{\Pr(S=k-3)}{\Pr(S=k-2)}.$$

for v and any $k\in\{2,\cdots,v\}$.

Hence, we have proven that inequality D.2 holds for $(v+1)$ and any $k\in\{2,\cdots,v\}$. To complete the proof, we also need to show that:

$$\frac{1}{v+1}\cdot\frac{\Pr(S'=v)}{\Pr(S'=v+1)}\geq\frac{1}{v}\cdot\frac{\Pr(S'=v-1)}{\Pr(S'=v)}$$

or

$$\Pr(S'=v)^2\geq\frac{v+1}{v}\cdot\Pr(S'=v-1)\Pr(S'=v+1) \qquad \text{(D.5)}$$

where $k = v + 1$ in inequalities D.2 and D.3.

Again by expanding $\Pr(S' = \cdot)$, we will find:

$$\begin{aligned}\Pr(S' = v)^2 &= (1 - q_{v+1})^2 \Pr(S = v)^2 + q_{v+1}^2 \Pr(S = v-1)^2 \\ &\quad + 2q_{v+1}(1 - q_{v+1}) \Pr(S = v) \Pr(S = v-1)\end{aligned}$$

and

$$\begin{aligned}\Pr(S' = k)\Pr(S' = k-2) &= (1 - q_{v+1})^2 \Pr(S = v+1)\Pr(S = v-1) \\ &\quad + q_{v+1}^2 \Pr(S = v)\Pr(S = v-2) \\ &\quad + q_{v+1}(1 - q_{v+1}) \times [\Pr(S = v)\Pr(S = v-1) \\ &\quad + \Pr(S = v+1)\Pr(S = v-2)] \\ &= q_{v+1}^2 \Pr(S = v)\Pr(S = v-2) \\ &\quad + q_{v+1}(1 - q_{v+1})\Pr(S = v)\Pr(S = v-1).\end{aligned}$$

since $\Pr(S = v + 1) = 0$.

Now the rest of the proof is trivial since we have:

$$\begin{aligned}\Pr(S = v-1)^2 &\geq \frac{v}{v-1} \cdot \Pr(S = v)\Pr(S' = v-2) \\ &> \frac{v+1}{v} \cdot \Pr(S = v)\Pr(S' = v-2)\end{aligned}$$

for $v \geq 2$.

By induction on v we have proven that:

$$\frac{1}{k} \cdot \frac{\Pr(S = k-1)}{\Pr(S = k)} > \frac{1}{k-1} \cdot \frac{\Pr(S = k-2)}{\Pr(S = k-1)} \tag{D.6}$$

for any value of $k \in \{1, 2, \cdots, v\}$ and thus:

$$\frac{\Pr(S = k-1)}{\Pr(S = k)} - \frac{\Pr(S = k-2)}{\Pr(S = k-1)} \geq \frac{1}{k-1} \cdot \frac{\Pr(S = k-2)}{\Pr(S = k-1)}$$

$$\begin{aligned}
&\geq \frac{1}{k-2} \cdot \frac{\Pr(S=k-3)}{\Pr(S=k-2)} \\
&\geq \cdots \\
&\geq \frac{\Pr(S=0)}{\Pr(S=1)} \\
&= \frac{\Pi_{i=1}^{v}(1-q_i)}{\Pi_{i\neq j}^{v}(1-q_i)\sum_{j=1}^{v} q_j} \\
&= \frac{1}{v < \alpha >_v}
\end{aligned}$$

Since Eq. D.6 is valid for any k, we can write:

$$\begin{aligned}
\frac{\Pr(S=k)}{\Pr(S=k-1)} &\geq \frac{k+1}{k} \cdot \frac{\Pr(S=k+1)}{\Pr(S=k)} \\
&\geq \frac{k+2}{k} \cdot \frac{\Pr(S=k+2)}{\Pr(S=k+1)} \\
&\geq \cdots \\
&\geq \frac{v}{k} \cdot \frac{\Pr(S=v)}{\Pr(S=v-1)} \\
&= \frac{v}{k} \cdot \frac{\Pi_{i=1}^{v} q_i}{\Pi_{i\neq j}^{v} q_i \sum_{j=1}^{v}(1-q_j)} \\
&= \frac{1}{k} \cdot \frac{1}{< 1/\alpha >_v} \\
&\geq \frac{1}{v < 1/\alpha >_v}
\end{aligned}$$

Therefore, the lemma is proven.

Appendix E

Proof of Lemma 3.4 in Chapter 3

We present the proof for the OR case. The AND case proof follows from this one by symmetry.

If a variable x_i is a child of an OR gate g, then recall that we can assume, w.l.o.g., that there must exists another variable x_j which is a descendent of g (otherwise x_i is a singleton).

In this case, we have that $\text{Infl}(x_j|x_i = 1) = \text{Infl}(g)\text{Infl}_g(x_j|x_i = 1) = 0$ and, consequently, $A^1_{j,i} = 0$.

Now, if x_i is not a child of an OR gate, we have:

$$\begin{aligned} A^1_{j,i} &= \frac{\text{Infl}(x_j|x_i = 1)}{\text{Infl}(x_j)} \\ &= \frac{\text{Infl}(x_j|x_i = 1)}{\text{Infl}(x_j|x_i = 1)\text{Pr}(x_i = 1) + \text{Infl}(x_j|x_i = 0)\text{Pr}(x_i = 0)} \\ &> \min\left\{1, \frac{\text{Infl}(x_j|x_i = 1)}{\text{Infl}(x_j|x_i = 0)}\right\} \end{aligned}$$

There exist two different cases.

Case 1: x_i and x_j be siblings to a perceptron g with a threshold θ. Then, let S be

the sum of g's children except x_i and x_j. We have:

$$\begin{aligned}\frac{\text{Infl}(x_j|x_i=1)}{\text{Infl}(x_j|x_i=0)} &= \frac{\Pr(S=\theta-1)}{\Pr(S=\theta)} \\ &> \gamma_n \qquad \text{from lemma 3.3}\end{aligned}$$

Case 2: x_i and x_j meet at a (higher level) perceptron g. Let g_i and g_j be the children of g through which x_i and x_j (respectively) feed g. Let θ be the threshold of g. Let $P^i_{ab} \stackrel{\text{def}}{=} \Pr(g_i = a|x_i = b)$. Let S be the sum of g's children except g_i and g_j. Then:

$$\begin{aligned}\frac{\text{Infl}(x_j|x_i=1)}{\text{Infl}(x_j|x_i=0)} &= \frac{\text{Infl}(g_j|g_i=1)P^i_{11}+\text{Infl}(g_j|g_i=0)P^i_{01}}{\text{Infl}(g_j|g_i=1)P^i_{10}+\text{Infl}(g_j|g_i=0)P^i_{00}} \\ &\geq \frac{\min\{\text{Infl}(g_j|g_i=1),\text{Infl}(g_j|g_i=0)\}}{\max\{\text{Infl}(g_j|g_i=1),\text{Infl}(g_j|g_i=0)\}} \\ &= \min\left\{\frac{\text{Infl}(g_j|g_i=1)}{\text{Infl}(g_j|g_i=0)},\frac{\text{Infl}(g_j|g_i=0)}{\text{Infl}(g_j|g_i=1)}\right\} \\ &= \min\left\{\frac{\Pr(S=\theta-1)}{\Pr(S=\theta)},\frac{\Pr(S=\theta)}{\Pr(S=\theta-1)}\right\} \\ &> \gamma_n \qquad \text{from lemma 3.3}\end{aligned}$$

Thus, $A^1_{j,i} > \gamma_n$ whenever x_i is not a child of an OR gate.

Appendix F

Proof of Lemma 3.5 in Chapter 3

The proof directly follows from the definition of $C^k_{i,j}$ and lemma 3.3. Here, we give the proof for the property 1; the proof for the rest is similar.

Let $\{x_i, x_j, x_k\}$ be a triple of variables siblings to an NG-perceptron g with a threshold θ and let $S \stackrel{\text{def}}{=} \sum_{l \neq i,j,k} x_l$ where x_l is a child of g. Then, we have:

$$
\begin{aligned}
C^k_{i,j} &= \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=0)} - \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)} \\
&+ \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)} - \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=0)} \\
&= \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)\mathrm{Infl}_g(x_j|x_i=0,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)\mathrm{Infl}_g(x_j|x_k=0)} \times \\
&\quad \left[\frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=0)}{\mathrm{Infl}_g(x_j|x_i=0,x_k=0)} - \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=1)}{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)}\right] \\
&= \frac{\Pr(S=\theta-1)\Pr(S=\theta)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)\mathrm{Infl}_g(x_j|x_k=0)} \times \\
&\quad \left[\frac{\Pr(S=\theta-1)}{\Pr(S=\theta)} - \frac{\Pr(S=\theta-2)}{\Pr(S=\theta-1)}\right] \\
&> \frac{\Pr(S=\theta-1)\Pr(S=\theta)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)\mathrm{Infl}_g(x_j|x_k=0)} \times \\
&\quad \frac{1}{\theta-1}\frac{\Pr(S=\theta-2)}{\Pr(S=\theta-1)} \qquad \text{from lemma 3.3}
\end{aligned}
$$

$$
\begin{aligned}
&> \frac{1}{\theta-1} \times \\
&\quad \frac{\Pr(S=\theta-2)\Pr(S=\theta)}{\max[\Pr(S=\theta-2),\Pr(S=\theta-1)]\cdot\max[\Pr(S=\theta-1),\Pr(S=\theta)]} \\
&> \gamma_n^2 \geq \omega \qquad \text{from lemma 3.3}
\end{aligned}
$$

Now, suppose that x_i, x_j, and x_k meet an NG-perceptron g. Let g_i, g_j and g_k be the children of g through which x_i, x_j and x_k (respectively) feed g. Then, we can show that:

$$
\begin{aligned}
C_{i,j}^k &= \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=0)} - \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)} \\
&+ \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)} - \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=0)} \\
&= \frac{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)\mathrm{Infl}_g(x_j|x_i=0,x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}_g(x_j|x_k=1)\mathrm{Infl}_g(x_j|x_k=0)} \times \\
&\quad \left[\frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=0)}{\mathrm{Infl}_g(x_j|x_i=0,x_k=0)} - \frac{\mathrm{Infl}_g(x_j|x_i=1,x_k=1)}{\mathrm{Infl}_g(x_j|x_i=0,x_k=1)}\right] \\
&= \frac{\mathrm{Infl}_{g_i}(x_i)\mathrm{Infl}_{g_k}(x_k)}{\mathrm{Infl}(x_i)} \times \\
&\quad \frac{\mathrm{Infl}_g(g_j|g_i=0,g_k=1)\mathrm{Infl}_g(g_j|g_i=0,g_k=0)}{\mathrm{Infl}_g(g_j|x_k=1)\mathrm{Infl}_g(g_j|x_k=0)} \times \\
&\quad \left[\frac{\mathrm{Infl}_g(g_j|g_i=1,g_k=0)}{\mathrm{Infl}_g(g_j|g_i=0,g_k=0)} - \frac{\mathrm{Infl}_g(g_j|g_i=1,g_k=1)}{\mathrm{Infl}_g(g_j|g_i=0,g_k=1)}\right]
\end{aligned}
$$

Let $S \stackrel{\text{def}}{=} \sum_{l\neq i,j,k} g_l$ where g_l is a child of g with the threshold θ. Then:

$$
\begin{aligned}
C_{i,j}^k &= \frac{\mathrm{Infl}_{g_k}(x_k)}{\mathrm{Infl}(g_i)} \cdot \frac{1}{\theta-1} \frac{\Pr(S=\theta-2)\Pr(S=\theta)}{\mathrm{Infl}_g(g_j|x_k=1)\mathrm{Infl}_g(g_j|x_k=0)} \\
&> \frac{\mathrm{Infl}_{g_k}(x_k)}{\mathrm{Infl}(g_i)} \cdot \frac{1}{\theta-1} \times \\
&\quad \frac{\Pr(S=\theta-2)\Pr(S=\theta)}{\max[\Pr(S=\theta-2)^2,\Pr(S=\theta-1)^2,\Pr(S=\theta)^2]} \\
&> \mathrm{Infl}_{g_k}(x_k)\gamma_n^2 \qquad \text{from lemma 3.3} \\
&\geq \epsilon_I \gamma_n^2 = \omega
\end{aligned}
$$

So this proves property 1.

Appendix G

Proof of Lemma 3.7 in Chapter 3

Here, we present the proof of Lemma 3.7 in Chapter 3. To prove this lemma, it is sufficient to show that any CMSLL V w.r.t. to a C-safe set W (in which there is no NG-CMSLL of three or more relevant variables) must satisfy conditions (1) and (2) of a G-CMSLL candidate. Indeed, if there exists another $x_k \in W - V$ such that (1) and (2) are satisfied for $U = V \cup \{x_k\}$, then U is a G-CMSLL candidate w.r.t. W and $V \subset U$.

Recall that, at this point, if V is a CMSLL w.r.t. W, then V is either an NG-CMSLL of only two variables or a G-CMSLL of two or more variables. For both of these cases, property 2 of Lemma 3.5 states that condition (1) is satisfied. For condition (2), we consider separately the case where V contains two variables and the case where V contains at least three variables.

i) The set $V = \{x_i, x_j\}$ is a G-CMSLL or an NG-CMSLL containing only two variables meeting at g. Then, for any $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$, we will have one and only one of the following cases:

(1) $\{x_i, x_l, x_m\}$ is meeting at g'. This implies that $\{g, x_l, x_m\}$ and $\{x_j, x_l, x_m\}$ are

also meeting at g' since x_i feeds g' through g and x_j feeds g. Then, from the definition of $C^k_{i,j}$, we have that $C^m_{i,l} = C^m_{g,l} = C^m_{j,l}$.

(2) $x_m \notin \text{desc}(\text{lca}(x_i, x_l))$. Note that $\text{lca}(x_i, x_l) = \text{lca}(x_j, x_l) = \text{lca}(g, x_l)$ since x_i and x_j feed x_l's parent through g. Then, $C^m_{i,l} = C^m_{j,l} = 0$.

(3) $x_l \notin \text{desc}(g_m)$ and g_m is $\text{lca}(x_i, x_m)$. Note again that $\text{lca}(x_i, x_m) = \text{lca}(x_j, x_m) = \text{lca}(g, x_m)$ since x_i and x_j feed x_m's parent through g. Then:

$$
\begin{aligned}
C^m_{i,l} &= \frac{\text{Infl}(x_l|x_i=1, x_m=0)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=0)} - \frac{\text{Infl}(x_l|x_i=1, x_m=1)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=1)} \\
&+ \frac{\text{Infl}(x_l|x_i=0, x_m=1)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=1)} - \frac{\text{Infl}(x_l|x_i=0, x_m=0)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=0)} \\
&= [\text{Infl}(x_l|g_m=1) - \text{Infl}(x_l|g_m=0)] \times \\
&\quad \left[\frac{\text{Infl}_{g_m}(x_i|x_m=0)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=0)} - \frac{\text{Infl}_{g_m}(x_i|x_m=1)}{\text{Infl}(x_i)\text{Infl}(x_l|x_m=1)}\right] \\
&= [\text{Infl}(x_l|g_m=1) - \text{Infl}(x_l|g_m=0)] \times \\
&\quad \left[\frac{\text{Infl}_{g_m}(g|x_m=0)}{\text{Infl}(g)\text{Infl}(x_l|x_m=0)} - \frac{\text{Infl}_{g_m}(g|x_m=1)}{\text{Infl}(g)\text{Infl}(x_l|x_m=1)}\right] \\
&= C^m_{g,l} = C^m_{j,l}.
\end{aligned}
$$

ii) The set $V = \{x_i, x_j, x_k, \cdots\}$ is a G-CMSLL containing three or more variables meeting at g. Since V is a C-safe set, each $x_i \in V$ must feed g through g_i (an NG-perceptron or a G-perceptron). Then, for any $\{x_l, x_m\} \subseteq W - \{x_i, x_j\}$, we first might have cases identical to those of (*i*) except that g now has at least three children $\{g_i, g_j, g_k\}$ fed correspondingly by $\{x_i, x_j, x_k\}$. For these cases, the same proofs as in (*i*) apply. In addition, we might have the following cases:

(1) $\{x_l, x_m\} \in V$. Then, we have that $C^m_{i,l} = C^m_{j,l} = 0$ from Property 2 of Lemma 3.5.

(2) $\{x_l\} \in V$ and $\{x_m\} \notin V$. Then, we have again that $C^m_{i,l} = C^m_{j,l} = 0$ from property 2 of Lemma 3.5.

(3) $\{x_l\} \notin V$ and $\{x_m\} \in V$. Then:

$$
\begin{aligned}
C_{i,l}^{m} &= \frac{\mathrm{Infl}(x_l|x_i=1,x_m=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=0)} - \frac{\mathrm{Infl}(x_l|x_i=1,x_m=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=1)} \\
&+ \frac{\mathrm{Infl}(x_l|x_i=0,x_m=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=1)} - \frac{\mathrm{Infl}(x_l|x_i=0,x_m=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=0)} \\
&= [\mathrm{Infl}(x_l|g=1) - \mathrm{Infl}(x_l|g=0)] \times \\
&\quad \left[\frac{\mathrm{Infl}_g(x_i|x_m=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=0)} - \frac{\mathrm{Infl}_g(x_i|x_m=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=1)}\right] \\
&= [\mathrm{Infl}(x_l|g=1) - \mathrm{Infl}(x_l|g=0)] \times \\
&\quad \left[\frac{\mathrm{Infl}_{g_i}(x_i)\mathrm{Infl}_g(g_i|g_m=0)\mathrm{Pr}(g_m=0|x_m=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=0)}\right. \\
&\quad \left. - \frac{\mathrm{Infl}_{g_i}(x_i)\mathrm{Infl}_g(g_i|g_m=0)\mathrm{Pr}(g_m=0|x_m=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_l|x_m=1)}\right] \\
&= [\mathrm{Infl}(x_l|g=1) - \mathrm{Infl}(x_l|g=0)] \times \\
&\quad \left[\frac{\mathrm{Pr}(g_m=0|x_m=0)}{\mathrm{Pr}(g_m=0)\mathrm{Infl}(x_l|x_m=0)} - \frac{\mathrm{Pr}(g_m=0|x_m=1)}{\mathrm{Pr}(g_m=0)\mathrm{Infl}(x_l|x_m=1)}\right] \\
&= C_{j,l}^{m}.
\end{aligned}
$$

Appendix H

Proof of Lemma 3.8 in Chapter 3

First note that if V contains only two variables, we trivially have (b) since (a) can only occur with a set of at least three variables. Hence, we now consider that V has at least three variables.

Let $V = \{x_i, x_j, x_k, \cdots\}$ be a G-CMSLL w.r.t. W containing three or more variables meeting at g. Let x_i and $x_j \in V$ feed g through its children g_i and g_j, respectively. We consider first g to be an AND gate. Then, for any $\{x_i, x_j\} \subset V$ we have:

$$\begin{aligned}
\Gamma_{i,j} &= \frac{A^1_{i,j} - A^0_{i,j}}{\mathrm{Infl}(x_j)} \\
&= \frac{\mathrm{Infl}(x_i|x_j = 1) - \mathrm{Infl}(x_i|x_j = 0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j)} \\
&= \frac{\mathrm{Infl}(g_i|g_j = 1) - \mathrm{Infl}(g_i|g_j = 0)}{\mathrm{Infl}(g_i)\mathrm{Infl}(g_j)} \\
&= \frac{\mathrm{Infl}(g_i|g_j = 1) - 0}{\mathrm{Infl}(g_i)\mathrm{Infl}(g_j)} \\
&> \frac{\mathrm{Infl}(g_i|g_j = 1)}{\mathrm{Infl}(g_i|g_j = 1)\mathrm{Pr}(g_j = 1)\mathrm{Infl}(g_j)} \\
&> \frac{1}{\mathrm{Pr}(g_j = 1)\mathrm{Infl}(g_j)}
\end{aligned}$$

$$> 1$$

If g is an OR gate, we have $\text{Infl}(g_i|g_j = 1) = 0$. In this case, we will find:

$$\Gamma_{i,j} < -1.$$

This proves that (a) is never held if V is a G-CMSLL; therefore, (b) is held.

Now, suppose that V is not a G-CMSLL w.r.t. W. Since V is a G-CMSLL candidate, the substructure emanating from the lca(V) has to be of a simple form. Indeed, for any $\{x_i, x_j\} \subset V$, the lca($\{x_i, x_j\}$) must be either an NG-perceptron of two (relevant) children or a G-perceptron of two or more children. If the lca($\{x_i, x_j\}$) is a G-perceptron, then (from the condition (1) for a G-CMSLL candidate) it cannot be fed by an NG-perceptron (of two relevant children). Hence, G-perceptrons occur only at the lowest level in the substructure emanating from the lca(V). For the rest of the proof, we consider separately the case where there exists at least one G-perceptron in the substructure emanating from the lca(V) from the case where there does not exist a G-perceptron in that substructure.

Suppose that there exists a G-perceptron in the substructure emanating from the lca(V). Since V, by hypothesis, is not a G-CMSLL and since a G-perceptron can only occur at the lowest level in the substructure emanating from the lca(V), then there exists $\{x_i, x_j, x_k\} \subseteq V$ such that the lca($\{x_i, x_j, x_k\}$) is an NG-perceptron and that the lca($\{x_i, x_k\}$) is a G-perceptron (an example is illustrated in Fig. 3.2 (c)). Let us consider that lca($\{x_i, x_k\}$) is an AND gate g; therefore, $\Gamma_{i,k} > 1$ (the OR gate case is similar but, for that, we have $\Gamma_{i,k} < -1$). Now, let x_i and $x_k \in V$ feed g through its children g_i and g_k, respectively. In this case, we find:

$$C^k_{i,j} = \frac{\text{Infl}(x_j|x_i = 1, x_k = 0) - \text{Infl}(x_j|x_i = 0, x_k = 0)}{\text{Infl}(x_i)\text{Infl}(x_j|x_k = 0)}$$

$$
\begin{aligned}
& - \frac{\mathrm{Infl}(x_j|x_i=1,x_k=1) - \mathrm{Infl}(x_j|x_i=0,x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=1)} \\
= \; & [\mathrm{Infl}(x_j|g=1) - \mathrm{Infl}(x_j|g=0)] \times \\
& \left\{ \frac{\mathrm{Infl}_g(x_i|x_k=0)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=0)} - \frac{\mathrm{Infl}_g(x_i|x_k=1)}{\mathrm{Infl}(x_i)\mathrm{Infl}(x_j|x_k=1)} \right\} \\
= \; & [\mathrm{Infl}(x_j|g=1) - \mathrm{Infl}(x_j|g=0)] \times \\
& \left\{ \frac{\mathrm{Infl}_g(g_i|g_k=0)\Pr(g_k=0|x_k=0)}{\mathrm{Infl}(g_i)\mathrm{Infl}(x_j|x_k=0)} \right. \\
& \left. - \frac{\mathrm{Infl}_g(g_i|g_k=0)\Pr(g_k=0|x_k=1)}{\mathrm{Infl}(g_i)\mathrm{Infl}(x_j|x_k=1)} \right\} \cdot \\
= \; & [\mathrm{Infl}(x_j|g=1) - \mathrm{Infl}(x_j|g=0)] \frac{\mathrm{Infl}_g(g_i|g_k=0)}{\mathrm{Infl}(g_i)} \times \\
& \left\{ \frac{\Pr(g_k=0|x_k=0)}{\mathrm{Infl}(x_j|x_k=0)} - \frac{\Pr(g_k=0|x_k=1)}{\mathrm{Infl}(x_j|x_k=1)} \right\} \\
= \; & [\mathrm{Infl}(x_j|g=1) - \mathrm{Infl}(x_j|g=0)] \frac{\mathrm{Infl}_g(g_i|g_k=0)}{\mathrm{Infl}(g_i)} \times \\
& \left\{ \frac{\mathrm{Infl}_{g_k}(x_k)\mathrm{Infl}(x_j|g_k=1)}{\mathrm{Infl}(x_j|x_k=0)\mathrm{Infl}(x_j|x_k=1)} \right\} \\
= \; & \frac{[\mathrm{Infl}(x_j|g=1) - \mathrm{Infl}(x_j|g=0)]}{\mathrm{Infl}(x_j)\mathrm{Infl}(g)} \frac{\mathrm{Infl}_g(g_i|g_k=0)}{\mathrm{Infl}_g(g_i)} \times \\
& \left\{ \frac{\mathrm{Infl}_{g_k}(x_k)\mathrm{Infl}(x_j|g_k=1)\mathrm{Infl}(x_j)}{\mathrm{Infl}(x_j|x_k=0)\mathrm{Infl}(x_j|x_k=1)} \right\} \\
= \; & \frac{\Gamma_{j,i}}{\Pr(g_k=0)} \left\{ \frac{\mathrm{Infl}_{g_k}(x_k)\mathrm{Infl}(x_j|g_k=1)\mathrm{Infl}(x_j)}{\mathrm{Infl}(x_j|x_k=0)\mathrm{Infl}(x_j|x_k=1)} \right\}.
\end{aligned}
$$

To have the condition (1) of Definition 3.2 satisfied for $\{x_i, x_j, x_k\}$, we need to have $C^k_{i,j} = 0$. However, since we will have access only to the empirical estimate of $C^k_{i,j}$, let us be conservative and consider that the condition (1) of Definition 3.2 is satisfied for $C^k_{i,j} \leq \omega$. The last equality tells us that to have $C^k_{i,j} \leq \omega$, it is necessary to have:

$$
\Gamma_{j,i} \cdot \min \left\{ \frac{\mathrm{Infl}_{g_k}(x_k)\mathrm{Infl}(x_j|g_k=1)\mathrm{Infl}(x_j)}{\Pr(g_k=0)\mathrm{Infl}(x_j|x_k=0)\mathrm{Infl}(x_j|x_k=1)} \right\} \leq \omega.
$$

Note that:

$$
\frac{\mathrm{Infl}(x_j)}{\mathrm{Infl}(x_j|x_k=0)\mathrm{Infl}(x_j|x_k=1)} \geq \min\left\{ \frac{1}{\mathrm{Infl}(x_j|x_k=0)}, \frac{1}{\mathrm{Infl}(x_j|x_k=1)} \right\}
$$

and

$$\min\{\frac{\text{Infl}(x_j|g_k=1)}{\text{Infl}(x_j|x_k=0)}, \frac{\text{Infl}(x_j|g_k=1)}{\text{Infl}(x_j|x_k=1)}\} \geq \min\{1, \frac{\text{Infl}(x_j|g_k=1)}{\text{Infl}(x_j|g_k=0)}\} > \gamma_n.$$

Therefore,

$$\frac{\text{Infl}_{g_k}(x_k)\text{Infl}(x_j|g_k=1)\text{Infl}(x_j)}{\Pr(g_k=0)\text{Infl}(x_j|x_k=0)\text{Infl}(x_j|x_k=1)} \geq \epsilon_I \cdot \gamma_n.$$

Since $\omega \stackrel{\text{def}}{=} \epsilon_I \gamma_n^2$, we find that $\Gamma_{j,i} \leq \gamma_n$. On the other hand, $\Gamma_{i,k} > 1$ (for an AND gate) and $\Gamma_{j,i} = \Gamma_{j,g} = \Gamma_{j,k}$ (from the definition of $\Gamma_{j,i}$). Here, $\Gamma_{i,k} \neq \Gamma_{i,j} = \Gamma_{j,k}$. So, this shows that the case (a) of the lemma is held here and, also, that x_j has been correctly identified as not feeding the $\text{lca}(\{x_i, x_k\})$.

Now, suppose that there does not exist a G-perceptron in the substructure emanating from the lca(V). Examples of this case are shown in Fig. 3.2 (b) and (d1). Because each NG-perceptron of two relevant variables has at least one irrelevant child, Lemma 3.2 implies that each of these perceptrons can be replaced in the hypothesis by either an OR or an AND gate without making much error. Moreover, we can find cases where case (a) would not hold, then this would be of no consequence since the whole substructure can be replaced by a single G-perceptron without making much error. In addition, we can also find cases where the case (a) holds. Then the only thing we need to guard against in these cases is the fact that we will always identify correctly the variable x_j which does not feed the $\text{lca}(\{x_i, x_k\})$. We can easily see that this will always be the case directly from the definition of $\Gamma_{j,i}$. Indeed, consider w.l.o.g. that $x_j \notin \text{desc}(\text{lca}(\{x_i, x_k\}))$, then (from the definition) we must always have $\Gamma_{j,i} = \Gamma_{j,k}$ so that it is only possible to have $\Gamma_{i,k} \neq \Gamma_{i,j} = \Gamma_{j,k}$.

Appendix I

Proof of Lemma 3.11 in Chapter 3

Here we are going to show that whenever $\tilde{f}$ is a stochastic function obtained from the target function f by replacing each perceptron g of the network by $\tilde{g}'$ where each $\tilde{g}'$ is obtained from g by the definitions given in 3.1 or 3.2, we then have:

$$\mathrm{err}(f, \tilde{f}) \leq 2 \sum_{i=1}^{r} \mathrm{Infl}(x_i) \mathrm{Pr}(x_i = 1) \mathrm{Pr}(x_i = 0)$$

where r is the total number of variables that have been removed.

To do so, let $W = \{G_1, \cdots, G_w\}$ be the set of children of f. Let $\tilde{G}_i$ be obtained from G_i by replacing its children by its approximated children as described in section 3.10. We divide W into two disjoint subsets V and U. Set V contains the G_is for which $\mathrm{err}(G_i, \tilde{G}_i) \leq \mathrm{Pr}(G_i = 1)$ and $\mathrm{err}(G_i, \tilde{G}_i) \leq \mathrm{Pr}(G_i = 0)$. Set U contains the G_is for which $\mathrm{Pr}(G_i = a) \leq \mathrm{err}(G_i, \tilde{G}_i) \leq \mathrm{Pr}(G_i = \bar{a})$ where $a \in \{0, 1\}$ and $\bar{a}$ is the negation of a.

Note that $\tilde{f}$ is a stochastic function obtained from f by replacing each perceptron $G_i \in W$ by $\tilde{G}_i'$ where each $\tilde{G}_i'$ is obtained from G_i by the definition given in 3.2. Therefore,

$$\mathrm{Pr}(\tilde{f} = 1 | S = l) = \mathrm{Pr}(f = 1 | S = l)$$

where $S \stackrel{\text{def}}{=} \sum_{i=1}^{|W|} \tilde{G}'_i$.

Let $\tilde{f}_V$ be the stochastic function obtained from f by ignoring all $G_i \in U$ and by replacing each perceptron $G_i \in V$ by $\tilde{G}'_i$. Therefore,

$$\Pr(\tilde{f}_V = 1 | S_V = l) = \Pr(f = 1 | S_V = l)$$

where $S_V \stackrel{\text{def}}{=} \sum_{i=1}^{|V|} \tilde{G}'_i$.

Now to prove the lemma, we first show that:

$$\text{err}(f, \tilde{f}) \leq \text{err}(f, \tilde{f}_V)$$

then, we show that:

$$\text{err}(f, \tilde{f}_V) \leq \sum_{i=1}^{|W|} \text{Infl}(G_i)\text{err}(G_i, \tilde{G}_i)$$

To prove that $\text{err}(f, \tilde{f}) \leq \text{err}(f, \tilde{f}_V)$, we can prove it w.l.o.g. for the case where $|U| = 1$.

$$\begin{aligned}
\text{err}(f, \tilde{f}) &= \Pr(f = 1, \tilde{f} = 0) + \Pr(f = 0, \tilde{f} = 1) \\
&= \sum_l \Pr(S = l)[\Pr(f = 1 | S = l)\Pr(\tilde{f} = 0 | S = l) \\
&\qquad + \Pr(f = 0 | S = l)\Pr(\tilde{f} = 1 | S = l)] \\
&= \Pr(f = 1) + \sum_l \Pr(S = l)\Pr(\tilde{f} = 1 | S = l) \\
&\qquad \times [1 - 2\Pr(f = 1 | S = l)] \\
&= \Pr(f = 1) + \sum_l \Pr(S = l)\Pr(f = 1 | S = l) \\
&\qquad \times [1 - 2\Pr(f = 1 | S = l)]
\end{aligned}$$

where, as before, $S = \sum_{i=1}^{|W|} \tilde{G}'_i$. Similarly:

$$\begin{aligned}
\text{err}(f, \tilde{f}_V) &= \Pr(f = 1) + \sum_l \Pr(S_V = l)\Pr(f = 1 | S_V = l) \\
&\qquad \times [1 - 2\Pr(f = 1 | S_V = l)]
\end{aligned}$$

where $S_{V'} \stackrel{\text{def}}{=} \sum_{i=1}^{|V|} \tilde{G}'_i$. Note that $S - S_{V'} = \tilde{G}'_1$ where $\tilde{G}'_1$ is obtained from $G_1 \in U$.

To show $\text{err}(f, \tilde{f}) \leq \text{err}(f, \tilde{f}_{V'})$, we show that $\text{err}(f, \tilde{f}) - \text{err}(f, \tilde{f}_{V'}) \leq 0$.

$$\begin{aligned}
&\text{err}(f, \tilde{f}) - \text{err}(f, \tilde{f}_{V'}) \\
&\quad = \textstyle\sum_l \Pr(S = l)\Pr(f = 1|S = l)[1 - 2\Pr(f = 1|S = l)] \\
&\quad\quad - \textstyle\sum_l \Pr(S_{V'} = l)\Pr(f = 1|S_{V'} = l)[1 - 2\Pr(f = 1|S_{V'} = l)] \\
&\quad = \textstyle\sum_l \Pr(S_{V'} = l)\{ \\
&\quad\quad \Pr(\tilde{G}'_1 = 1)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 1)[1 - 2\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 1)] \\
&\quad\quad +\Pr(\tilde{G}'_1 = 0)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 0)[1 - 2\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 0)] \\
&\quad\quad -[\Pr(\tilde{G}'_1 = 1)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 1) \\
&\quad\quad\quad +\Pr(\tilde{G}'_1 = 0)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 0)] \times \\
&\quad\quad [1 - 2\Pr(\tilde{G}'_1 = 1)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 1) \\
&\quad\quad\quad -2\Pr(\tilde{G}'_1 = 0)\Pr(f = 1|S_{V'} = l, \tilde{G}'_1 = 0)]\} \\
&\quad = -2\Pr(\tilde{G}'_1 = 1)\Pr(\tilde{G}'_1 = 0) \textstyle\sum_l \Pr(S_{V'} = l)[\text{Infl}(\tilde{G}'_1|S_{V'} = l)]^2 \\
&\quad \leq 0
\end{aligned}$$

We now prove that $\text{err}(f, \tilde{f}_{V'}) \leq \sum_{i=1}^{|V'|} \text{Infl}(G_i)\text{err}(G_i, \tilde{G}_i)$. For this, note that since $\Pr(f = 1|\mathbf{x})$ is either zero or one, we can write:

$$\text{err}(f, \tilde{f}_{V'}) \stackrel{\text{def}}{=} \sum_{\mathbf{x}} \Pr(\mathbf{x})|\Pr(f = 1|\mathbf{x}) - \Pr(\tilde{f}_V = 1|\mathbf{x}_V)|$$

where $\mathbf{x}$ is the setting of all the variables, $\mathbf{x}_V$ is the setting of the relevant variables that meet each $G_i \in V'$, and the sum is over all the possible values of $\mathbf{x}$.

Using the same techniques as for Lemmas 3.9 and 3.10, we can show that:

$$\text{err}(f, \tilde{f}_{V'}) \leq \sum_{i=1}^{|U|} \text{Infl}(G_i)\Pr(G_i = 1)\Pr(G_i = 0) + \sum_{i=1}^{|V|} \text{Infl}(G_i)\text{err}(G_i, \tilde{G}'_i)$$

Note that $G_i \in U$ in the first term and that $G_i \in V$ in the second term.

Since $\text{err}(G_i, \tilde{G}_i) \leq \Pr(G_i = 1)$ and $\text{err}(G_i, \tilde{G}_i) \leq \Pr(G_i = 0)$ for all $G_i \in V$, we have that $\text{err}(G_i, \tilde{G}'_i) \leq \text{err}(G_i, \tilde{G}_i)$ for all $G_i \in V$. Also, we have:

$$\Pr(G_i = 1)\Pr(G_i = 0) < \text{err}(G_i, \tilde{G}_i)$$

for all $G_i \in U$.

Therefore,

$$\mathrm{err}(f, \tilde{f}_V) \leq \sum_{i=1}^{|W|} \mathrm{Infl}(G_i)\mathrm{err}(G_i, \tilde{G}_i)$$

and thus

$$\mathrm{err}(f, \tilde{f}) \leq \sum_{i=1}^{|W|} \mathrm{Infl}(G_i)\mathrm{err}(G_i, \tilde{G}_i).$$

We can perform the same analysis for each G_i, we thus find:

$$\mathrm{err}(G_i, \tilde{G}_i) \leq \sum_{j=1}^{|W_i|} \mathrm{Infl}_{G_i}(g_j)\mathrm{err}(g_j, \tilde{g}_j).$$

where g_j is a child of G_i and continuing this to the bottom level perceptrons for which we have:

$$\mathrm{err}(g, \tilde{g}) \leq 2 \sum_{i} \mathrm{Infl}_g(x_i)\mathrm{Pr}(x_i = 1)\mathrm{Pr}(x_i = 0),$$

we will find:

$$\mathrm{err}(f, \tilde{f}) \leq 2 \sum_{i=1}^{r} \mathrm{Infl}(x_i)\mathrm{Pr}(x_i = 1)\mathrm{Pr}(x_i = 0)$$

where now x_i belongs to the set of all removed variables and r denotes the total number of removed variables.

Appendix J

Estimation of m in Chapter 3

The number m of required examples is such that all the probabilities given in the proof of theorem 3.1 must be estimated with the given precision and confidence. This will be the case if we choose m to satisfy:

$$\Pr\left\{|\hat{C}_{i,j}^{k} - C_{i,j}^{k}| < \frac{\epsilon_I \gamma_n^2}{4}\right\} > 1 - \frac{\delta}{8n^3}$$

which is equivalent to:

$$\Pr\left\{\left|\frac{\hat{\text{Infl}}(x_j|x_i=1,x_k=0)}{\hat{\text{Infl}}(x_i)\hat{\text{Infl}}(x_j|x_k=0)} - \frac{\text{Infl}(x_j|x_i=1,x_k=0)}{\text{Infl}(x_i)\text{Infl}(x_j|x_k=0)}\right| < \frac{\epsilon_I \gamma_n^2}{16}\right\} > 1 - \frac{\delta}{32n^3}\,.$$

Let $I_1 \stackrel{\text{def}}{=} \text{Infl}(x_j|x_i=1,x_k=0)$, $I_2 \stackrel{\text{def}}{=} \text{Infl}(x_j)$, $I_3 \stackrel{\text{def}}{=} \text{Infl}(x_j|x_k=0)$, $\epsilon' \stackrel{\text{def}}{=} (\epsilon_I \gamma_n^2)/16$, and $\delta' \stackrel{\text{def}}{=} \delta/(32n^3)$. Now, since we have:

$$\begin{aligned}
\left|\frac{\hat{I}_1}{\hat{I}_2\hat{I}_3} - \frac{I_1}{I_2 I_3}\right| &= \left|\frac{\hat{I}_1 - I_1}{\hat{I}_2\hat{I}_3} - \frac{I_1(\hat{I}_2\hat{I}_3 - I_2 I_3)}{\hat{I}_2\hat{I}_3 I_2 I_3}\right| \\
&\leq \frac{|\hat{I}_1 - I_1|}{\hat{I}_2\hat{I}_3} + \frac{I_1|\hat{I}_2\hat{I}_3 - I_2 I_3|}{\hat{I}_2\hat{I}_3 I_2 I_3} \\
&\leq \frac{|\hat{I}_1 - I_1|}{\hat{I}_2\hat{I}_3} + \frac{I_1|\hat{I}_2 - I_2|}{\hat{I}_2 I_2 I_3} + \frac{I_1|\hat{I}_3 - I_3|}{\hat{I}_2\hat{I}_3 I_3}\,,
\end{aligned}$$

in order to have:

$$\Pr\left\{\left|\frac{\hat{I}_1}{\hat{I}_2\hat{I}_3}-\frac{I_1}{I_2I_3}\right|<\epsilon'\right\}>1-\delta'\ , \tag{J.1}$$

it is sufficient to have:

$$\Pr\left\{\frac{|\hat{I}_1-I_1|}{\hat{I}_2\hat{I}_3}<\frac{\epsilon'}{2}\right\}>1-\frac{\delta'}{3}\ ,\ \Pr\left\{\frac{I_1|\hat{I}_2-I_2|}{\hat{I}_2I_2I_3}<\frac{\epsilon'}{4}\right\}>1-\frac{\delta'}{3}\ ,$$

$$\text{and}\quad \Pr\left\{\frac{I_1|\hat{I}_3-I_3|}{\hat{I}_2\hat{I}_3I_3}<\frac{\epsilon'}{4}\right\}>1-\frac{\delta'}{3}\ .$$

These are equivalent to have:

$$\Pr\left\{|\hat{I}_1-I_1|<\frac{\epsilon'}{2}\cdot\hat{I}_2\hat{I}_3\right\}>1-\frac{\delta'}{3}\ ,\ \Pr\left\{|\hat{I}_2-I_2|<\frac{\epsilon'}{4}\cdot\frac{\hat{I}_2I_2I_3}{I_1}\right\}>1-\frac{\delta'}{3},$$

$$\text{and}\quad \Pr\left\{|\hat{I}_3-I_3|<\frac{\epsilon'}{4}\cdot\frac{\hat{I}_2\hat{I}_3I_3}{I_1}\right\}>1-\frac{\delta'}{3}\ .$$

Hence, we now have the following bounds:

$\hat{I}_2 > 3\epsilon_I/2$ in order to have $I_2 > \epsilon_I$, $I_3 > \gamma_n\epsilon_I$, $\hat{I}_2\hat{I}_3 > (9\gamma_n\epsilon_I^2)/4$ in order to have $I_3 > \epsilon_I\gamma_n$, $(I_2I_3)/I_1 > \gamma_n$(from Lemma3.3), and $(\hat{I}_2\hat{I}_3)/I_1 > 3\gamma_n/4$. Hence, to satisfy inequality J.1, it is sufficient to have:

$$\Pr\left\{|\hat{I}_1-I_1|<\frac{\epsilon'}{2}\cdot\frac{9\gamma_n\epsilon_I^2}{4}\right\}>1-\frac{\delta'}{3}\ ,\ \Pr\left\{|\hat{I}_2-I_2|<\frac{\epsilon'}{4}\cdot\frac{3\epsilon_I\gamma_n}{2}\right\}>1-\frac{\delta'}{3},$$

$$\text{and}\quad \Pr\left\{|\hat{I}_3-I_3|<\frac{\epsilon'}{4}\cdot\frac{3\epsilon_I\gamma_n^2}{4}\right\}>1-\frac{\delta'}{3}\ .$$

Let $\epsilon_I \le \gamma_n/6$. In that case, satisfying the first inequality, implies that the second and third inequalities will be satisfied.

Now, to have:

$$\Pr\left\{|\hat{I}_1-I_1|<\frac{\epsilon_I\gamma_n^2}{16}\frac{9\gamma_n\epsilon_I^2}{8}\right\}>1-\frac{\delta}{96n^3}$$

we need to have:

$$\Pr\left\{\left|\hat{\Pr}(f=1|x_j=x_i=1,x_k=0)-\Pr(f=1|x_j=x_i=1,x_k=0)\right|<\frac{9\epsilon_I^3\gamma_n^3}{256}\right\}$$

$$> 1 - \frac{\delta}{192n^3} .$$

Following Pagallo and Haussler [24], to satisfy this inequality, it is sufficient to have:

$$\Pr\left\{\left|\hat{\Pr}(x_j=x_i=1, x_k=0) - \Pr(x_j=x_i=1, x_k=0)\right| < \frac{9\epsilon_I^3\gamma_n^3}{256} \cdot \frac{\epsilon_S^3}{4}\right\}$$

$$> 1 - \frac{\delta}{384n^3}$$

Using the multiplication form of the Chernoff bounds [11]:

$$\Pr\{|\hat{p} - p| > \gamma \cdot p\} < 2e^{-m\gamma^2 p/2} \leq \delta,$$

we find that, to satisfy our last inequality it is sufficient to have:

$$m \geq \frac{1}{3}\left(\frac{6}{\epsilon_S}\right)^3 \left(\frac{8}{3\epsilon_I\gamma_n}\right)^6 \ln\left(\frac{12(4n)^3}{\delta}\right) .$$

Bibliography

[1] Anthony M., (1997) "Probabilistic Analysis of learning in Artificial Neural Networks: The PAC Model and its Variants", *Neural Computing Surveys* vol. 1, 1–47.

[2] Abend K., Hartley T.J. & Kanal L.N. (1965) "Classification of Binary Random Patterns", *IEEE Trans. Inform. Theory* vol. IT-11, 538–544.

[3] Auer P., Herbster M., Warmuth M., (1996) "Exponentially Many Local Minima for Single Neurons", in *Advances in Neural Processing Systems 8*, Touretzky D.S., Mozer M.C., Hasselmo M.E. eds, 316–322. MIT Press.

[4] Baum E., Hassler D., (1989) "What Size Net Gives Valid Generalization?", *Neural Computation*, vol. 1, 151–160.

[5] Block H.D. (1962) "The Perceptron: A Model for Brain Functioning", *Reviews of Modern Physics*, vol. 34, 123–135.

[6] Blum A. & Rivest R.L. (1988). Training a 3-node neural network is NP-complete'. In *Proc. of the 1st Workshop on Computational Learning Theory*, p. 9. Morgan Kaufmann.

[7] Golea, M., Marchand M. (1993) "On Learning Perceptrons with Binary Weights", *Neural Computation* vol. 5, 765–782.

[8] Golea M., and Marchand M., (1993) "Polynomial Time Algorithms for Learning Neural Nets of Nonoverlapping Perceptrons" *Computational Intelligence* vol. 9, pp. 155–170.

[9] Golea, M. & Marchand M. (1994) "On Learning Simple Deterministic and Probabilistic Neural Concepts", to appear in the *Proceedings of EuroCOLT'93*, Oxford University Press.

[10] Golea M., Marchand M., & Hancock T.R., (1996) "On Learning μ-Perceptron Networks On the Uniform Distribution", *Neural Networks* vol. 9, 67–82.

[11] Hagerup T. & Rub C., (1989) "A Guided Tour to Chernoff Bounds", *Info. Proc. Lett.*, Vol. 33, 305–308.

[12] Hancock T.R., Golea M., & Marchand M., (1994) "Learning Nonoverlapping Perceptron Networks from Examples and Membership Queries", *Machine Learning*, vol. 16, pp. 161–183.

[13] Haussler D. (1992) "Decision Theoritic Generalizations of the PAC Model for Neural Net and Other Learning Applications", *Information and Computation* vol. 100, 78–150.

[14] Hoeffgen K.U. (1993) "On Learning and Robust Learning of Product Distributions", *Proceedings of the 6th ACM Conference on Computational Learning Theory*, ACM Press, 77–83.

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

[16] Ibragimov I.A., (1956) "On the composition of unimodal distributions", *Theor. Probability Appl.* vol. 1, 255–266. *Also:* Keilson J. & Gerber H., (1971) "Some results for discrete unimodality", *J. Amer. Statist. Assoc.*, vol. 66, 386–389.

[17] Kearns M.J., Li M., Pitt L. & Valiant L. G., (1987). "On the learnability of boolean formulae" *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing* (pp. 285–295). New York: ACM Press.

[18] Kearns M. (1993). "Efficient Noise-Tolerant Learning from Statistical Queries", *Proceedings of the Twenty Fifth Annual ACM Symposium on the Theory of Computation*, p. 392.

[19] Kearns M. and Schapire R.E. (1990) " Efficient Distribution-free Learning of Probabilistic Concepts", in *Proceedings of the 31st Symposium on Foundations of Computer Science*, pp. 382.

[20] Kearns M.J. and Schapire R.E. (1994) " Efficient Distribution-free Learning of Probabilistic Concepts", *Journal of Computer and System Sciences*, Vol. 48, pp. 464–497.

[21] Lin J.H. & Vitter J.S. (1991) "Complexity Results on Learning by Neural Nets", *Machine Learning*, Vol. 6, 211-230.

[22] Marchand M., and Hadjifaradji S., (1995) "Learning Stochastic Perceptrons under k-Blocking Distributions", in G. Tesauro, D. S. Touretzky and T. K. Leen, eds., *Advances in Neural Information Processing Systems 7*, pp. 279–286, MIT Press, Cambridge MA.

[23] Marchand M. and Hadjifaradji S., (1996) "Strong Unimodality and Exact Learning of Constant Depth μ-Perceptron Networks", in D. S. Touretzky, M. C. Mozer and

M. E. Hasselmo, eds., *Advances in Neural Information Processing Systems 8*, pp. 288–294, MIT Press, Cambridge MA.

[24] Pagallo G. and Haussler D. (1989) "A Greedy Method for Learning μ DNF fucntion under the Uniform Distribution", *Technial Report* UCSC-CRL-89-12.

[25] Rosenblatt F., (1958) "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Phychological Review*, vol. 65, 386–407.

[26] Rumelhart D.E., Hinton G.E., Williams R.J. (1986) "Learning Internal Representations by Error Backpropagation", in *Parallel Distributed Processing 1*, Rumelhart D.E, McClelland J.L. eds, 318–362, MIT Press.

[27] Ripley B.D, (1996). *Pattern Recognition and neural Networks*, Cambridge University Press.

[28] Schapire R.E. (1992) *The Design and Analysis of Efficient Learning Algorithms*, Cambridge MA: MIT Press.

[29] Schapire R., (1994) "Learning probabilistic read-once formulas on product distributions" *Machine Learning* vol. 14, 47–81. San Mateo, CA: Morgan Kaufman.

[30] Valiant L.G. (1984) "A Theory of the Learnable", *Comm. ACM*, Vol. 27, 1134–1142.

[31] Yamanishi K. (1992) "A Learning Criteria for Stochastic Rules", *Machine Learning*, vol. 9, 165–203.

[32] Vapnik V.N. (1982) *Estimation of Dependences Based on Empirical Data*, Springer.

[33] Vapnik V.N. (1995) *The Nature of Statistical Learning Theory*, Springer.

[34] Vapnik V.N. (1998) *Statistical Learning Theory*, Wiley.

[35] Werbos P.J., (1974) "Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Boston MA.