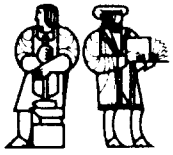
LABORATORY FOR
COMPUTER SCIENCE



MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-391

EFFICIENT METHODS FOR CALCULATING MAXIMUM ENTROPY DISTRIBUTIONS

Sally A. Goldman

May 1987

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

This blank page was inserted to preserve pagination.

Efficient Methods for Calculating Maximum Entropy Distributions

by

Sally A. Goldman
Sc.B., Brown University
(1985)

Submitted to the Department of
Electrical Engineering and Computer Science
in Partial Fulfillment of the
Requirements of the Degree of
Master of Science in
Electrical Engineering and Computer Science
at the
Massachusetts Institute of Technology
March 1987

Signature of Author _____
Department of Electrical Engineering and Computer Science
March 11, 1987

Certified by _____
Ronald L. Rivest
Professor of Computer Science and Engineering

Accepted by _____
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Efficient Methods for Calculating Maximum Entropy Distributions

Sally A. Goldman

Submitted to the Department of Electrical Engineering and Computer Science on March 11, 1987 in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering and Computer Science

Abstract

We present a new algorithm for computing the maximum entropy probability distribution satisfying a set of constraints. Unlike previous approaches, our method is integrated with the planning of data collection and tabulation. We show how *adding constraints* and performing the associated additional tabulations can substantially speed up computation by replacing the usual iterative techniques with a straight-forward computation. We note, however, that the constraints added may contain significantly more variables than any of the original constraints so there may not be enough data to collect meaningful statistics. These extra constraints are shown to correspond to the intermediate tables in Cheeseman's method. Furthermore, we prove that acyclic hypergraphs and decomposable models are equivalent, and discuss the similarities and differences between our algorithm and Spiegelhalter's algorithm. Finally, we compare our work to Kim and Pearl's work on singly-connected networks.

Portions of this thesis are joint work with Ronald Rivest.

Keywords: maximum entropy, uncertain reasoning, probabilistic techniques, acylic hypergraphs, constrained optimization.

Thesis Supervisor: Professor Ronald L. Rivest

Title: Professor of Computer Science and Engineering

This research was supported in part by NSF Grants DCR-8006938 and DCR-8607494, and in part by an Office of Naval Research Fellowship.

Acknowledgments

I would like to thank my thesis advisor, Ron Rivest. Without his guidance this thesis would not have been possible. He provided me with many useful insights and helped me to clarify my ideas.

I would also like to thank Tom Cormen, Ray Hirschfeld, Cindy Phillips, Bob Sloan, and Mike Wellman for many helpful discussions. I would like to give special thanks to Tom Cormen for all the help he gave me with LaTeX.

Finally, this thesis could not have been completed without all the support I received from my husband, Ken. Really, he should have been included in my list of colleagues because he spent many hours helping me with the technical aspects of my thesis. He provided me with many useful comments on several drafts of this thesis. And most of all, when I needed him, he was there for me.

Contents

Chapter 1

Introduction

Many applications require reasoning with incomplete information. For example, one may wish to develop expert systems that can answer questions based on an incomplete model of the world. Having an incomplete model means that some questions may have more than one answer consistent with the model. How can a system choose reasonable answers to these questions?

Many solutions to this problem have been proposed. While probability theory is the most widely used formalism for representing uncertainty, ad-hoc approximations to probability have been used in practice. The problem with pure probabilistic approaches is that their computational complexity seems prohibitive. One way to reduce the complexity of this problem is to make the strong assumption of conditional independence. However, this assumption is typically not valid and generates inaccurate results. A probabilistic method which shows promise for solving some of the problems with uncertain reasoning is the maximum entropy approach.

This thesis discusses efficient techniques, based on the principle of maximum entropy, for answering questions when given an incomplete model. The organization is as follows. In the remainder of this chapter, we formally define the inference problem and justify the use of the maximum entropy principle. In chapter 2, previously known methods for calculat-

ing the maximum entropy distribution are discussed. Then, in chapter 3, we present a new technique which makes maximum entropy computations easier by adding extra constraints, and we discuss an alternative approach which leads to another efficient algorithm for calculating the maximum entropy probability distribution. In chapter 4, we compare our new technique to some of the other techniques discussed in the thesis. Finally, chapter 5 contains conclusions and discussion of open problems.

1.1 Formal Problem Definition

In this section, we formally define the inference problem which this thesis addresses. We begin by defining some notation. Let $V = \{A, B, C, \ldots\}$ be a finite set of binary-valued variables, or attributes. (The generalization to finite-valued variables is straightforward.) Consider the event space Ω_V defined to be the set of all mappings from V to $\{0, 1\}$. We call such mappings *assignments* since they assign a value to each variable in V. It is easy to see that $|\Omega_V| = 2^{|V|}$. If $E \subseteq V$, we have Ω_V is isomorphic to $\Omega_E \times \Omega_{V-E}$; we identify assignments in Ω_E with subsets of Ω_V in the natural manner.

We are interested in probability distributions defined on Ω_V. We use the following convention throughout this paper. If $E \subseteq V$, we write $P(E)$ to denote the probability of an element of $\Omega_E \subseteq \Omega_V$. In other words, we specify only the variables involved in the assignments and not their values. For example,

$$P(V) = P(A)P(B)P(C)\cdots \tag{1.1}$$

represents $2^{|V|}$ equations, stating that the variables are independent. (We do not assume equation (1.1).) By convention, all assignments in an equation must be consistent. We also write $P(A)$ instead of $P(\{A\})$, $P(AB)$ instead of $P(\{AB\})$, and so on.

We use a similar convention for summations: $\sum_E$ stands for a summation over all assignments in Ω_E, when $E \subseteq V$. Using these conventions, we see

that $Y \subseteq E \subseteq V$ implies that

$$P(Y) = \sum_{E-Y} P(E) \tag{1.2}$$

For example, if $E = \{ABCD\}$ and $Y = \{AB\}$ then $P(Y) = \sum_{CD} P(E)$.

For conditional probabilities we use similar notation. For $Y \subseteq E \subseteq V$ the probability of E given Y is written as $P(E|Y)$ and defined to be

$$P(E|Y) = P(E - Y|Y) = \frac{P(E \wedge Y)}{P(Y)}. \tag{1.3}$$

We say that X and Y are *conditionally independent* given S if

$$P(X \wedge Y|S) = P(X|S)P(Y|S) \tag{1.4}$$

where $X, Y, S \subseteq V$.

We are interested in probability distributions on Ω_V satisfying a set of constraints. We assume that the constraints are supplied in the form of *joint marginal probabilities.* Let $E_1, \ldots, E_m$ be distinct but not necessarily disjoint subsets of V. Let us suppose that for each i we are given the $2^{|E_i|}$ constraint values $\{P(E_i)\}$. Furthermore, we assume that these values are *consistent.* By consistent we mean that there exists at least one probability distribution on Ω_V which satisfies the constraints. Note that equation (1.2) states that a constraint on the values $P(Y)$ is implied by a constraint on the values $P(E)$ when $Y \subseteq E$. A common way of ensuring that the constraints are consistent is to derive the constraints by computing the observed marginal probabilities from a common set of data [1].

Many techniques require that constraints are given in the form of *conditional probabilities.* Here, each element of $E_1, \ldots, E_m$ has a distinguished subset e_i, and the input is the $2^{|E_i|}$ constraint values $\{P(E_i - e_i|e_i)\}$ for each i. This approach is frequently used when obtaining information form

[1] Using "experts" to provide subjective probability estimates is a well-known way of deriving a set of *inconsistent* constraints [36].

experts because it is often easier for experts to give information in terms of conditional probabilities. In general, we find that joint marginal distributions are easier to handle. Since we plan to obtain the constraint values from raw data, we are free to use joint marginal distributions. By doing so, we avoid both the possibility of inconsistent data, and the difficulty in transforming conditional distributions to joint marginal distributions.

In general, there may be many probability distributions satisfying the constraints. There are two problems which must be addressed. First, assuming that there are many probability distributions satisfying the constraints, which one should be chosen? And second, how can one *efficiently* calculate the desired distribution? Most of our attention shall be given to the second of these two problems, but we briefly address the first in the following section.

1.2 The Maximum Entropy Principle

In this section, we formally define the maximum entropy principle and summarize arguments justifying its use. When faced with an underconstrained problem, a reasonable way to get a unique answer is to apply the principle of maximum entropy. The entropy function, H, is defined as follows:

$$H(P) = -\sum_{V} P(V)\log(P(V)). \tag{1.5}$$

The maximum entropy probability distribution, P^*, is the unique distribution which maximizes H while satisfying the supplied constraints. Motivation for this choice are given by Jaynes, Rissanen, Shore and Johnson, and Tikochinsky, Tishby and Levine [17,18,19,27,30,35]. We summarize their arguments in the following two sections.

1.2.1 Informal Justification

In this section, we provide intuitive arguments as to why the probability distribution that maximizes the entropy function is the appropriate choice to make when dealing with an underconstrained problem.

Informally, the maximum entropy principle says that when one makes inferences based on incomplete information, one should draw them from the probability distribution that has the maximum uncertainty permitted by the data. That is, the maximum entropy distribution is the unique distribution which is maximally noncommittal with regard to missing information. There are many intuitive reasons why probability distributions of high entropy are favored over others [17,18]. Some such reasons are that the distributions of higher entropy assume less, are more probable, and are smoother. From an information theoretic viewpoint, the possible distributions are concentrated near the one of maximum entropy. That is, given incomplete information, the maximum entropy distribution is not only realized in the greatest number of ways, but for large N the overwhelming majority of all *possible* distributions compatible with the data have entropy very close to the maximum. Thus to choose an estimate other than the one that maximizes entropy would amount to ignoring the vast majority of all the possibilities allowed by the data, and concentrating on a small and unrepresentative subclass of them.

1.2.2 Formal Justification

While the informal justification of the previous section provides a convincing argument for applying the maximum entropy principle, one does not have to rely on these intuitive arguments. In this section, we discuss some formal arguments for choosing the probability distribution which maximizes entropy.

We would like any general method of inference to obtain identical solu-

tions if the same input is processed in different yet equivalent ways. Since the maximum entropy principle is designed as a general method of inference, one would like to show that it maintains consistency. Johnson and Shore [30,19] prove the even stronger result that the principle of maximum entropy is the only method of inference which satisfies a set of consistency properties. They prove that the principle of maximum entropy is correct in the following sense: maximizing any function but entropy will lead to inconsistency unless that function and entropy have identical maxima. They give the following set of consistency axioms:

1. *Uniqueness:* The result should be unique.
2. *Invariance:* The choice of coordinate system should not matter.
3. *System Independence:* It should not matter whether one accounts for independent information about independent systems separately in terms of different densities or together in terms of a joint density.
4. *Subset Independence:* It should not matter whether one treats an independent subset of system states in terms of a separate conditional density or in terms of the full system density.

Then Johnson and Shore prove that if the input is in the form of constraints on expected values, the unique distribution which satisfies the constraints and the consistency axioms is the one obtained by maximizing the entropy function.

Some very interesting results which are also based on satisfying a consistency property are provided by Tikochinsky, Tishby, and Levine [35]. They assume the event space is Ω_V and the constraints are m expected values $E(i)$ where $i = 1, \ldots, m$. The goal is to find a probability distribution P which fulfills

$$E(i) = \sum_V P(V)A(i,V) \tag{1.6}$$

These constraints are more general than the constraints discussed in section 1.1. If we restricted $A(i,V)$ to be either 0 meaning element V should not be included or 1 meaning element V should be included, and $E(i)$ to be the sum of the probabilities of the included elements, then we get constraints which are joint marginal probability distributions.

Now assume that assignment v occurred N_v times in N independent trials. Let $\tilde{N} = (N_1, \ldots, N_{|\Omega_V|})$ be any particular distribution and define

$$E'(i) = \sum_{\tilde{N}} \left(\sum_{V} N_V A(i,V) \right) \tag{1.7}$$

In other words, $E'(i)$ is the expected number of occurrences of event i if N independent (reproducible) trials were performed. Now there are two ways to compute $P_{\tilde{N}}$:

1. Apply some algorithm (a) to the constraints E to get P and then $P_{\tilde{N}} = (\prod_V P(V)^{N_V})N!/(\prod_V N_V!)$.

2. First apply equation (1.7) to get E' and then apply algorithm (a) to E' to get $P_{\tilde{N}}$.

They define algorithm (a) to be consistent if these two routes yield the same results. And finally, they prove that algorithm (a) is consistent if and only if algorithm (a) is the maximum entropy procedure, thus justifying the maximum entropy principle.

Finally, one other way in which the maximum entropy principle can be justified is to justify a more general technique. Rissanen's work [27] provides this type of justification. His work deals with the principle of minimum description length (MDL). The MDL principle is based on minimizing the total number of binary digits required to rewrite the observed data. He justifies the MDL principle by arguing that it correctly expresses one's initial ignorance. He shows that when the parameters determine the data the MDL

principle degenerates to the principle of maximum entropy. Therefore, one could argue that in such a case maximizing entropy is desirable.

1.3 Log–Linear Representations

Provided that the maximum entropy distribution is the one of choice, we would like an algorithm that calculates it efficiently. The remaining chapters of this thesis will address this issue. One problem that is immediately apparent is that space required to store a probability distribution is exponential in $|V|$. In this section, we discuss an efficient way to store the maximum entropy probability distribution.

One advantage of the maximum entropy distribution, P^*, is that it has a simple representation. For each ω in Ω_{E_i}, there is a non-negative real parameter $\alpha_{E_i}(\omega)$ (i.e., one parameter set per constraint set and $2^{|E_i|}$ parameters in the parameter set for E_i), that determine P^* as follows. Let us write $\alpha_i(\omega)$ instead of $\alpha_{E_i}(\omega)$ for brevity, and omit the argument ω when it can be deduced from context. Now we may simply write

$$P^*(V) = \alpha_1 \alpha_2 \ldots \alpha_m. \tag{1.8}$$

Each element of Ω_V is assigned a probability which is the product of the appropriate α's where each α determines its argument from the assignment to V. This is known as a *log-linear* representation. For example, suppose we have the variables A, B, C and constraint sets $E_1 = \{AB\}$ and $E_2 = \{BC\}$. Then we will have the variable sets α_{AB} and α_{BC}, where the corresponding variables are $\alpha_{AB}(00), \ldots, \alpha_{AB}(11)$ and $\alpha_{BC}(00), \ldots, \alpha_{BC}(11)$. The maximum entropy distribution is given by the log–linear model

$$P^*(V) = \alpha_{AB}\alpha_{BC}.$$

If $P_{ABC}(010)$ (i.e., the probability that $A = 0$, $B = 1$, and $C = 0$) is desired, it can be calculated by

$$P^*(010) = \alpha_{AB}(01)\alpha_{BC}(10).$$

Thus, the maximum entropy distribution can be represented in a linear amount of space in the number of constraints.

Chapter 2

Previous Work

In the previous chapter, we argued that the desired probability distribution is the one which maximizes the entropy, and discussed a way to efficiently store the maximum entropy distribution. The essential question which remains is: How can we efficiently calculate the maximum entropy distribution? The next two chapters will address this question.

In principle, the problem of computing the maximum entropy distribution is solved by the classical Lagrange multipliers method. That is, we want to maximize equation (1.5) subject to constraints of the form

$$\sum_V P(V)A(i,V) = b(i) \quad \text{where } i = 1, \ldots, m. \tag{2.1}$$

These constraints are of the general form introduced in section 1.2.2 when discussing Tikochinsky, Tishby and Levine's work. Now, applying the Lagrange multiplier method we find that the maximum entropy distribution is given by

$$P^*(V) = \frac{1}{Z} \exp\left(-\sum_{i=1}^{m} \lambda_i A(i,V)\right) \tag{2.2}$$

where

$$Z = \sum_V \exp\left(-\sum_{i=1}^{m} \lambda_i A(i,V)\right) \tag{2.3}$$

and the λ_i's are determined by the constraints. (We note that the α's defined in section 1.3 come from the Lagrange multiplier technique, in fact $\alpha_i = e^{-\lambda_i}$.) Thus, calculating P^* is just the problem of solving a set of m non–linear equations to obtain $\lambda_1, \ldots, \lambda_m$ (or equivalently $\alpha_1, \ldots, \alpha_m$). Mathematically this problem has been solved with such techniques as Newton-Raphson, modified Newton-Raphson, and the conjugate gradient method [33]. So, in principle, we know how the calculate the maximum entropy distribution, but these techniques are iterative and require exponential time. Our goal is to find a more efficient algorithm to obtain the maximum entropy distribution.

In section 2.1, we discuss iterative methods to solve the set of non–linear equations for the α's. While in the general case it is necessary to solve for the α's to obtain the maximum entropy distribution, if appropriate restrictions are made then the maximum entropy distribution can be obtained without iteration. In section 2.2, some non–iterative techniques are introduced.

2.1 Iterative Maximum Entropy Methods

Most existing methods for calculating the maximum entropy distribution are iterative. They typically begin with a representation of the uniform distribution and converge towards a representation of the maximum entropy distribution. Each step adjusts the representation so that a given constraint is satisfied. To enforce a constraint $P(E_i)$, all of the elementary probabilities $P(V)$ relevant to that constraint are multiplied by a common factor. Because constraints are dependent, adjusting the representation to satisfy one constraint may cause a previously satisfied constraint to no longer hold. Thus, one must iterate repeatedly through the constraints until the desired accuracy is reached. (We note that the implicit constraint — that the probabilities sum to one — must usually be explicitly considered.) Examples of this type of algorithm are discussed in [5,6,13,16,20,22].

Representing the probability distribution explicitly as a table of $2^{|V|}$ values is usually impractical. For this problem, it is most convenient to store only $\alpha_1, \alpha_2, \ldots \alpha_m$; this is a representation as compact as the input data, which represents the current probability distribution implicitly via equation (1.8). To represent the uniform distribution, every α is set to 1, except for the α corresponding to the requirement that entries of the probability distribution must sum to 1 — which is set to $2^{-|V|}$. To determine if a constraint is satisfied, one must sum the appropriate elements of the probability distribution. Any particular element can be computed using equation (1.8). If the constraint is not satisfied, the relevant α is multiplied by the ratio of the desired sum to the computed sum. Thus, in originally calculating the α's and later in evaluating queries it is necessary to evaluate a sum of terms, where each term is a product of α's. This sum is difficult to compute since it may involve an exponential number of terms.

Cheeseman [6] proposes a clever technique for rewriting such sums in order to evaluate them more efficiently. For example

$$\alpha \sum_{A \ldots F} \alpha_{AB}\ \alpha_{ACD}\ \alpha_{DE}\ \alpha_{AEF}$$

is rewritten as follows. First, $\sum_{A \ldots F}$ is broken into six sums, each over one variable. Arbitrarily choosing the variable ordering $CDFEAB$, we obtain

$$\alpha \sum_B \sum_A \sum_E \sum_F \sum_D \sum_C \alpha_{AB} \alpha_{ACD} \alpha_{DE} \alpha_{AEF}.$$

Now each α is moved left as far as possible (it stops when reaching a sum over a variable on which it depends). The above sum then becomes

$$\alpha \sum_B \sum_A \alpha_{AB} \sum_E \sum_F \alpha_{AEF} \sum_D \alpha_{DE} \sum_C \alpha_{ACD}.$$

The sums are evaluated from right to left. The result of each sum is an intermediate *table* containing the value of the sum evaluated so far as a function of variables further to the left which have been referenced. For example

after evaluating the innermost summation a table, t_{AD}, is kept containing $\sum_C \alpha_{ACD}$ for Ω_{AD}. Then after evaluating the next sum, a table, t_{AE}, is kept containing $\sum_D \alpha_{DE} t_{AD}$ for Ω_{AE}. We continue in this manner until all the summations are evaluated. For this example, the 193 multiplications required by the naive method are reduced to only 21 multiplications. As we shall see, in some cases the intermediate table is most efficiently represented as two or more smaller tables. The variable ordering must be chosen carefully in order to take full advantage of this technique. A poor choice of variable ordering can yield a sum which is not much better than explicitly considering all $2^{|V|}$ terms; a good choice may dramatically reduce the work required. While picking a good variable ordering is important, the success of this technique depends greatly on the interconnectedness of the constraints. If the constraints are highly connected, no ordering can significantly reduce the complexity of evaluating the summation.

Some alternative approaches to the standard iterative schemes have been proposed. One of the more interesting proposals is due to Geman [14,23]. Instead of considering one constraint at a time, this algorithm uses stochastic relaxation to simultaneously adjust the probability distribution to meet all of the constraints. In particular, a convex function, whose minimum gives the maximum entropy distribution, is calculated. Then a technique to approximate the gradient and a gradient descent algorithm are used to find this minimum.

An approach which comes immediately from the Lagrange multiplier technique is discussed by Agmon, Alhassid, and Levine [1,2]. First they calculate the "potential function"

$$F(\lambda^t) = \log(Z(\lambda^t)) + \sum_{i=1}^{m} \lambda_i^t b(i) \tag{2.4}$$

where Z is as defined in equation (2.3). They show that F is strictly convex, and has a unique global minimum for λ^t which solves $\nabla F(\lambda^t) = 0$.

Finally, they use the modified Newton-Raphson procedure to find the global minimum.

Unlike most approaches which are based on the Lagrange multiplier technique, Csiszár [9] uses I–divergence geometry to prove that a generalized version of the standard iterative technique converges. His proof is more general than those which are derived from the Lagrange multiplier technique.

2.2 Non–Iterative Maximum Entropy Techniques

The iterative techniques of the previous section are applicable in most situations, but computationally they are rather inefficient. We want a model that is powerful enough to handle real–world situations, yet simple enough for the maximum entropy distribution to be calculated efficiently. In this section, we begin by discussing some non–iterative approaches which use conditional independence instead of the principle of maximum entropy to obtain a unique result. Then we introduce several non–iterative maximum entropy algorithms which will be discussed in more detail in chapter 3.

Chow and Liu [8] consider the class of product approximations in which only second–order distributions are used. If there is a product approximation such that for some ordering of the variables $x_1 \ldots x_n$, each x_i depends on at most one variable from the set $\{x_1, \ldots x_{i-1}\}$, then this approximation forms a *dependence tree.* They discuss how to build the best dependence tree when supplied with the complete probability distribution. They also present a method to construct an optimal dependence tree from samples.

Similar to their work is the work of Kim and Pearl [21]. They construct a *Bayesian network* where the nodes represent variables and directed links represent direct dependencies; all direct influences on a node come from its parents. We will use the following notation for stating their formula: S_x is the set of the *immediate* predecessors (parents) of node x in the network, T_x is the set of *all* predecessors of node x in the network, and $\mathcal{R}$ is the set

of roots (sources). All conditional probabilities of the form $P(x|S_x)$ for all $x \notin \mathcal{R}$ and $P(x)$ for all $x \in \mathcal{R}$, along with the independence assumptions that $P(x|S_x) = P(x|T_x)$ suffice to define the following unique probability distribution:

$$P^{\bullet}(V) = \left(\prod_{x \in \mathcal{R}} P(x) \right) \left(\prod_{x \notin \mathcal{R}} P(x|S_x) \right). \tag{2.5}$$

They define a network to be *singly-connected* if there is at most one *undirected* path between any pair of nodes. One of the most interesting results of this work is that the propagation of new evidence through a singly–connected network can be accomplished by a network of parallel processors in time proportional to the longest path in the network. Pearl [26] addresses the problem of propagating new evidence through *multiply–connected* networks.

Dalkey [10] has shown that if a Bayesian network is a tree (i.e. all nodes have at most one parent), equation (2.5) gives the maximum entropy distribution. So for a Bayesian network which is a tree, a non–iterative technique exists for calculating the maximum entropy distribution when the input is all conditional probabilities of the form $P(x|S_x)$ for all $x \notin \mathcal{R}$ and $P(x)$ for all $x \in \mathcal{R}$.

Let's now return to the maximum entropy approach. Even with Cheeseman's summation technique, the general iterative algorithm of section 2.1 has a very high (exponential) computational cost since many iterations through the constraints are still required before the distribution converges. The non–iterative techniques discussed up to this point are efficient, but they assume conditional independence which is rarely present. We want a non–iterative maximum entropy algorithm in order to reduce the time to compute the probability distribution. If we are willing to put restrictions on the supplied constraints, then this goal can be achieved.

Malvestuto [24] provides sufficient conditions for the existence of a non–iterative technique when the constraints are joint marginal probabilities. The details are discussed in section 3.1. Our work is based on the model

which Malvestuto introduced. We show that by *enlarging* the set of constraints to be considered *before* the data is gathered and tabulated, the difficulties of computing the maximum entropy distribution are substantially alleviated. That is, with the enlarged set of constraints, the computation of the maximum entropy distribution is non–iterative. The details of our method are discussed in section 3.2. Darroch, Lauritzen and Speed [11] also give sufficient conditions for the existence of a non–iterative technique when the constraints are joint probabilities. The details of their work are discussed in section 3.3. Edwards and Kreiner [12] and Wermuth and Lauritzen [37] extend this work. Some work which is similar to ours is that of Spiegelhalter [31,32]. He shows how to take any Bayesian network which has no *directed* cycles and convert it to an undirected representation which meets the restrictions of Darroch, Lauritzen and Speed.

Chapter 3

Efficient Maximum Entropy Algorithms

In this chapter, we cover two non–iterative maximum entropy algorithms. In section 3.1, we introduce a way to model constraints which are joint marginal probability distributions as a hypergraph, and present a non–iterative formula for the maximum entropy distribution for hypergraphs which are acyclic. In practice, this formula is not very useful since typically constraint sets do not form acyclic hypergraphs. In section 3.1, we propose a new maximum entropy algorithm which is based on the observation that a hypergraph can be made acyclic by *adding* hyperedges. In other words, a maximum entropy computation can actually be simplified by adding constraints. In section 3.3, we discuss an alternative way to graphically model the constraints and give a non–iterative formula for the maximum entropy distribution for models which are decomposable. Finally, in section 3.4, we describe Spiegelhalter's algorithm for estimating a probability distribution.x

3.1 Acyclic Hypergraphs

Our approach is based on the work of Malvestuto [24], who derived sufficient conditions for writing marginals of the maximum entropy distribution as a

product of easily calculated probabilities. In this section we introduce his work. We begin by describing how to model a set of variables and associated constraints as a hypergraph. It is interesting to note that the work on acyclic hypergraphs first appeared in the database literature [3,4,25]. The variables in our problem replace the attributes of the database, and the constraint sets replace the relations. If the database schema is acyclic, many problems can be simplified.

A hypergraph is like an ordinary undirected graph, except that each edge is an arbitrary subset of the vertices, instead of just a subset of size two. We define the hypergraph $H = (\mathcal{V}, \mathcal{E})$ to contain a vertex for each variable, and a hyperedge for each constraint. For example the hyperedge $\{ABC\}$ corresponds to the constraint set $E_i = \{A, B, C\}$. We say that hyperedge X *subsumes* hyperedge Y if $Y \subseteq X$. It is important to observe that the constraints on a sub-hypergraph induced by restricting attention to a subset of the vertices can be inferred from the original hypergraph constraints using equation (1.2).

We define the graph $C(H)$ of a hypergraph H to be the graph whose vertices are those of H and whose edges are the vertex pairs $\{v, w\}$ such that v and w are in a common hyperedge of H. A hypergraph H is *conformal* if every clique of $C(H)$ is contained in a hyperedge of H. A graph G is triangulated if for every cycle of length greater than three, there is an edge of G joining two non-consecutive vertices in the cycle. Such an edge is called a *chord* of the cycle; hence triangulated graphs are sometimes called *chordal* graphs. A hypergraph H is *acyclic* if H is conformal and $C(H)$ is chordal.

An equivalent definition is that a hypergraph is acyclic if repeatedly applying the following reduction steps results in the empty hypergraph (containing no edges and no vertices):

1. Delete any vertices which belong to only one hyperedge.

2. Delete any hyperedges which are subsumed by another hyperedge.

Graham's algorithm is the procedure of applying reduction steps 1 and 2 until either the empty set is reached, or neither can be applied [15].

Before proceeding, we shall define some notation regarding the above reduction procedure. Let $\mathcal{E}^{(0)} = \{E_1^{(0)}, \ldots, E_m^{(0)}\}$, where $E_i^{(0)}$ is the i^{th} hyperedge of H. Let $Y_i^{(k)}$ be the set of variables which appear in at least one hyperedge other than $E_i^{(k)}$. Finally let $\mathcal{E}^{(i+1)}$ be the result of applying reduction step (1) and then (2) to $\mathcal{E}^{(i)}$. If H is acyclic then there exists an l such that $\mathcal{E}^{(l+1)} = \emptyset$.

For acyclic hypergraphs, Malvestuto [24] gave the following formula for the maximum entropy distribution, $P^*(V)$:

$$P^*(V) = \left(\prod_{k=0}^{l-1} \frac{\prod_i P\left(E_i^{(k)}\right)}{\prod_i P\left(Y_i^{(k)}\right)} \right) \left(\prod_i P\left(E_i^{(l)}\right) \right) \tag{3.1}$$

Note that no α's are needed; the formula depends only on probabilities in the original input data (constraints). This formula is an immediate extension of the following theorem due to Malvestuto [24].

Theorem 1 *Given the constraints $E_1, \ldots, E_m$, the maximum entropy distribution is given by the following.*

$$P^*(V) = \frac{P(E_1) \cdots P(E_m)}{P(Y_1) \cdots P(Y_m)} P^*(Y)$$

where Y_i is the set of variables which appear in at least one hyperedge other than E_i and $P^*(Y)$ is the maximum entropy distribution for the constraints $Y_1, \ldots, Y_m$.

Proof: From the joint marginal constraints we have the following

$$\begin{aligned} P(E_i) &= \sum_{V - E_i} \alpha_1 \cdots \alpha_m \\ &= \alpha_i \sum_{V - E_i} \prod_{j \neq i} \alpha_j \end{aligned} \tag{3.2}$$

Similarly we have,

$$\begin{aligned} P(Y_i) &= \sum_{Z_i} \sum_{V-E_i} \alpha_1 \cdots \alpha_m \\ &= \left(\sum_{Z_i} \alpha_i \right) \left(\sum_{V-E_i} \prod_{j \neq i} \alpha_j \right) \end{aligned} \tag{3.3}$$

Let $\beta_i = \sum_{Z_i} \alpha_i$. Combining equations (3.2) and (3.3) from above gives:

$$\alpha_i = \frac{P(E_i)}{P(Y_i)} \beta_i \tag{3.4}$$

Now writing $P^*(V)$ in its product form we get

$$\begin{aligned} P^*(V) &= \alpha_1 \cdots \alpha_m \\ &= \frac{P(E_1) \cdots P(E_m)}{P(Y_1) \cdots P(Y_m)} \beta_1 \cdots \beta_m \end{aligned} \tag{3.5}$$

We want to show that $\psi(Y) = \beta_1 \cdots \beta_m$ is $P^*(Y)$, the maximum entropy distribution for the constraints $P(Y_i)$. To do this, it is suffices to prove that the joint marginal constraints hold.

$$\begin{aligned} P^*(E_i) &= \sum_{V-E_i} \left(\prod_j \frac{P(E_j)}{P(Y_j)} \right) \psi(Y) \\ &= \sum_{V-E_i} \psi(Y) \frac{P(E_i)}{P(Y_i)} \prod_{j \neq i} \frac{P(E_j)}{P(Y_j)} \\ &= \frac{P(E_i)}{P(Y_i)} \sum_{Y-Y_i} \left(\psi(Y) \prod_{j \neq i} \frac{1}{P(Y_j)} \sum_{Z-Z_i} \left(\prod_{j \neq i} P(E_j) \right) \right) \end{aligned} \tag{3.6}$$

where $Z = Z_1 \cup \cdots \cup Z_m$, so that $V = Y \cup Z$. Now, since the Z_j's are disjoint,

$$\begin{aligned} \sum_{Z-Z_i} \prod_{j \neq i} P(E_j) &= \prod_{j \neq i} \sum_{Z-Z_i} P(E_j) \\ &= \prod_{j \neq i} \sum_{Z_j} P(E_j) \\ &= \prod_{j \neq i} P(Y_j) \end{aligned} \tag{3.7}$$

Substituting equation (3.7) into equation (3.6) gives:

$$P^*(E_i) = \frac{P(E_i)}{P(Y_i)} \sum_{Y-Y_i} \psi(Y)$$

However since $P^*(E_i) = P(E_i)$ we get $P(Y_i) = \sum_{Y-Y_i} \psi(Y)$, so $\psi(Y)$ satisfies the constraints $P(Y_i)$. ■

3.2 A New Maximum Entropy Method

In this section, we propose a new procedure for calculating the maximum distribution. While Malvestuto's work provides a non–iterative maximum entropy algorithm for acyclic hypergraphs, in practice, this technique is not very useful since typically constraint sets do not form acyclic hypergraphs. While one could make a hypergraph acyclic by removing some hyperedges, this approach would lead to inaccurate results. In this section, we propose a new maximum entropy algorithm which is based on the observation that a hypergraph can be made acyclic by *adding* hyperedges. In other words, maximum entropy computations can actually be simplified by adding constraints. The main advantage of our procedure is that it avoids the iteration previously required by providing a non–iterative formula for the desired answer. The major disadvantage is that the method cannot ordinarily be applied if the data is already tabulated and the constraints already derived; the method requires that one "plan ahead" and tabulate additional constraints when processing the data.

3.2.1 Description

We begin by describing our algorithm. Equation (3.1) allows one to avoid iteration when calculating the maximum entropy distribution for schemas having acyclic hypergraphs. What should one do for *cyclic* hypergraphs? Our method is based on the observation that *a hypergraph can always be*

$$\{\underline{A}B, \underline{A}C\underline{D}, \underline{DE}, \underline{AE}F\}$$
$$\Downarrow$$
$$\{\underline{AD}, \underline{DE}, \underline{AE}\}$$

Figure 3.1: The hypergraph consisting of the hyperedges $\{AB\}$,$\{ACD\}$,$\{DE\}$, and $\{AEF\}$ is cyclic as shown be the reduction above. (Elements of Y_i are underlined.)

$$\{\underline{A}B, \underline{A}C\underline{D}, \underline{ADE}, \underline{AE}F\}$$
$$\Downarrow$$
$$\{ADE\}$$
$$\Downarrow$$
$$\emptyset$$

Figure 3.2: The hypergraph consisting of the hyperedges $\{AB\}$, $\{ACD\}$, $\{ADE\}$, and $\{AEF\}$ is acyclic as shown be the reduction above. (Elements of Y_i are underlined.)

made acyclic by adding hyperedges. This claim is trivial to prove, since at worst a hypergraph can be made acyclic by adding the hyperedge containing all vertices. For example, the hypergraph:

$$(\mathcal{V}, \mathcal{E}) = (\{ABCDEF\}, \{\{AB\}, \{ACD\}, \{DE\}, \{AEF\}\})$$

becomes acyclic when the hyperedge $\{ADE\}$ is added (see figures 3.1 and 3.2).

Thus, by adding additional constraints (edges) the maximum entropy calculation can be simplified so that no iteration is required. Here is a summary of how our method works:

1. We begin with a set of variables (attributes) and constraint sets deemed to be of interest. (Cheeseman [7] discusses a learning program which uses the raw data to find a set of significant constraints. Edwards

and Kreiner [12] also discuss how to choose a good set of constraints.) Here a "constraint set" is a set of variables; the intent is that during data-gathering there will be one joint marginal distribution table created for each constraint set, and the observed events will be tabulated once in each table according to the values of the attributes in the constraint set. For example, if $\{A, B, C\}$ is a constraint set of three binary–valued attributes, then there will be a table of size 8 used to categorize the data with respect to these three attributes. This results in 8 constraints on the maximum-entropy distribution desired, one for each of the eight observed probabilities $P(ABC)$.

2. Construct the corresponding hypergraph $H = (\mathcal{V}, \mathcal{E})$, where there is one vertex for each variable and one hyperedge corresponding to each constraint group.

3. Perform Graham's algorithm on H, and let H' denote the resulting hypergraph. If H' is the empty hypergraph, then H is acyclic, and the following step is skipped.

4. Find a set $\mathcal{X}$ of *additional* hyperedges (constraint groups) which can be added to H' to make it acyclic. Note that any original edges subsumed by edges in $\mathcal{X}$ are eliminated. These additional hyperedges should be chosen to minimize the space required to store the joint marginal distributions.

5. Collect data for the expanded set $\mathcal{E} \cup \mathcal{X}$ of constraints [1].

6. Apply equation (3.1) to calculate individual elements of the maximum entropy distribution.

[1] Our method is unusual in that it extends the set of tables (constraints) used to tabulate the data. To fill in the entries of a new table, the raw data must still be available in step (5). So, steps 1-4 may be considered to be "planning" steps.

With the exception of step 4, we have completely described how to perform each of the steps. We now discuss how to find a good set of hyperedges to add to make a hypergraph acyclic. Finding the optimal set of hyperedges to add is extremely similar to the minimum fill–in problem encountered when performing Gaussian elimination on sparse symmetric matrices [28,29]. Since the minimum fill–in problem has been proven to be NP–complete [38], we conjecture that our problem is as well. There are many heuristics which have been studied for the minimum fill–in problem. We plan on using one of these heuristics, the minimum–degree heuristic, to find a good set of hyperedges to add. Here is our proposal for performing step 4 of our algorithm. We define the *degree* of a vertex v to be the number of vertices in a common hyperedge with v. We begin by calculating the degree of each vertex in H'. Let v be the vertex with the smallest degree (break ties at random). Add to $\mathcal{X}$ the hyperedge e which contains v and any vertices in a common hyperedge with v. Next, modify H' by adding e to it and performing Graham's algorithm. If H' is now the empty hypergraph then we are done, otherwise return to the step of calculating the degree of each vertex. The reason for choosing this algorithm, is that it usually keeps the hyperedges in $\mathcal{X}$ as small as possible.

3.2.2 Example

In this section, we demonstrate our algorithm on the example of figure 3.3. First we must perform Graham's algorithm on H. The result is shown below, where elements of Y_i are underlined.

$$\{\underline{AB}, \underline{AC}, \underline{BCE}, \underline{BDF}, \underline{CD}\}$$
$$\Downarrow$$
$$\{\underline{AB}, \underline{AC}, \underline{BC}, \underline{BD}, \underline{CD}\}$$

So after performing step 3 of our algorithm we have the hypergraph H' shown in figure 4. Now we must find the set $\mathcal{X}$ of hyperedges to make H' acyclic. We start by calculating the degree of the vertices in H'. We

Figure 3.3: The original hypergraph H.

Figure 3.4: The hypergraph H' obtained by performing Graham's algorithm on the hypergraph H of figure 3.3.

find that A and D both have a degree of two, and B and C both have a degree of three. Since A (we could have picked D) has the smallest degree we let $\mathcal{X} = \{\{ABC\}\}$. After adding $\{ABC\}$ to H' and performing Graham's algorithm, we find that $H' = (\{BCD\}, \{\{BC\}, \{BD\}, \{CD\}\})$. Since H' is a complete graph the only way to make it acyclic is to add a hyperedge which contains all vertices of H'. Thus, $\mathcal{X} = \{\{ABC\}, \{BCD\}\}$. Finally after adding the hyperedges in $\mathcal{X}$ to those in H and removing any hyperedges subsumed by another, we obtain the acyclic hypergraph $(\{ABCDEF\}, \{\{ABC\}, \{BCD\}, \{BCE\}, \{BDF\}\})$. Therefore, the set of

constraints for which the data must be collected is $E_1 = \{ABC\}$, $E_2 = \{BCD\}$, $E_3 = \{BCE\}$, and $E_4 = \{BDF\}$. Now in order to apply equation (3.1) we must perform Graham's algorithm on this new set of constraints.

$$\begin{array}{c} \{A\underline{BC}, \underline{BCD}, \underline{BC}E, \underline{BD}F\} \\ \Downarrow \\ \{BCD\} \\ \Downarrow \\ \emptyset \end{array}$$

Now applying equation (3.1) we get the the maximum entropy distribution is as follows:

$$\begin{aligned} P^*(V) &= \frac{P(ABC)}{P(BC)} \frac{P(BCE)}{P(BC)} \frac{P(BDF)}{P(BD)} P(BCD) \\ &= \frac{P(ABC)P(BCE)P(BDF)P(BCD)}{P(BC)P(BC)P(BD)} \end{aligned} \tag{3.8}$$

3.2.3 Possible Problems

In this section, we consider possible inefficiencies of our method. First, it may be necessary to add "large" hyperedges containing many vertices in order to make the hypergraph acyclic. For example, to make the complete undirected graph (containing all hyperedges of size two) acyclic, one must add the "maximum" hyperedge containing all vertices. Since the size of the table corresponding to a hyperedge is an exponential function of the size of the hyperedge, adding large hyperedges creates a problem. Furthermore, the table corresponding to the maximum hyperedge is itself the probability distribution that we are estimating, so the above situation is clearly undesirable. This kind of behavior depends on the structure of the hypergraph; hypergraphs which are "highly connected" will tend to require the addition of large hyperedges. However, when the graph is highly connected other techniques seem to "blow up" as well.

Finally, because of our method's unique approach, we have a unique concern. Recall that since the data is tabulated *after* adding the additional

constraints; steps 1-4 of our algorithm must be performed while the source of the constraints (i.e., the raw data) is still available. If the added hyper-edges are too large, there may not be enough data to calculate meaningful statistics. Tabulating 100,000 data points in a table of size approximately 1,000 will give reasonable estimates, while tabulating them in a table of size approximately 1,000,000 will not.

3.3 Decomposable Models

In this section, we introduce the model discussed by Darroch, Lauritzen and Speed [11]. They prove that if the model corresponding to the constraints is a "decomposable model" then a non–iterative algorithm exists to calculate the maximum entropy distribution. We begin with some definitions. Given a set of joint marginal constraints $E_1, \ldots, E_m$ one can construct an undirected graph $G = (V, E)$ by having one vertex per variable, and putting an edge between any two vertices where the variables corresponding to the vertices are in the same constraint set. For example the graph

$$G = (\{ABC\}, \{\{AB\}, \{BC\}, \{AC\}\})$$

corresponds to the constraint set $E_i = \{A, B, C\}$. A *clique* is a set of vertices where each vertex in the set is connected to all other vertices in the set. A *maximal clique* is a clique which can not be extended by the addition of more vertices. The *generating class* for G is $C = \{A_1, \ldots, A_k\}$ where $A_1, \ldots, A_k$ are the maximal cliques of the graph G. A graph G is said to be a *graphical model* if every clique of G is contained in some E_i for $i = 1, \ldots, m$. (In other words, the members of the generating class for a graphical model are equivalent to the constraints which form the model.) For example, for the constraints $\{ABE\}$, $\{ADE\}$ and $\{ACD\}$ (see figure 3.5), the generating class is $\{ABE, ADE, ACD\}$. Therefore, this example is a graphical model. However, for the constraints $\{AB\}$, $\{BC\}$, and $\{AC\}$ (see figure 3.6) the

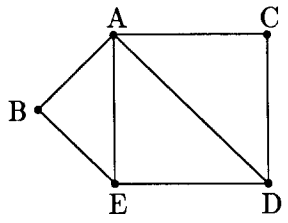


Figure 3.5: Example of a graphical model: ABE, ADE, ACD.

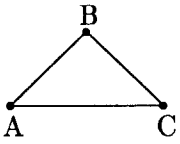


Figure 3.6: Example of a non-graphical model: AB, BC, AC.

generating class is $\{ABC\}$. This example is not a graphical model since $\{ABC\}$ is not a subset of $\{AB\}$, $\{BC\}$, or $\{AC\}$. Recall that a graph G is triangulated if for every cycle of length greater than three, there is an edge of G joining two non-consecutive vertices in the cycle. Finally, a *decomposable model* is a graphical model which is *triangulated.*

Before looking at the formula which Darroch, Lauritzen and Speed discuss for the maximum entropy distribution of a decomposable model, we need some more definitions and results. If a model is decomposable its joint distribution can be expressed in terms of an ordering of the constraints $E_1, \ldots, E_m$. Since we are dealing with decomposable models, this is equivalent to expressing the joint distribution in terms of an ordering of the maximal cliques $C = \{A_1, \ldots A_k\}$. We shall denote $V_t = E_1 \cup \cdots \cup E_t$. For example, $V_m = V$. There exists orderings such that $E_t \cap V_{t-1} = c_t \subseteq E_{r_t}$ for some $r_t \in \{1, \ldots, t-1\}$ where $2 \leq t \leq m$. Any ordering which obeys the

above property is said to have the *running intersection property.* This leads to the following formula for the maximum entropy distribution:

$$P^*(V_t) = P(E_1)\left(\prod_{r=2}^{t}\frac{P(E_r)}{P(c_r)}\right) \qquad t = 2,\ldots,m \tag{3.9}$$

3.4 Spiegelhalter's Approach

In this section, we describe a method independently introduced by Spiegelhalter [31,32] for calculating the maximum entropy distribution. He takes advantage of the fact that the joint distribution for decomposable models may be expressed as a simple function of the joint probabilities on the cliques and the clique intersections. The essence of his technique is to create a decomposable model from a *causal graph* which is the same as the Bayesian network of Kim and Pearl discussed in section 2.2.

We begin with Spiegelhalter's notation. Given a causal graph $G_d = (V, E)$ and $v \in V$, let D_v denote the set of direct descendants of v, and A_v denote the set of nodes not reachable from v. Spiegelhalter makes the following assumptions:

1. The causal graph has no directed cycles,
2. for all $v \in V : P(v|A_v) = P(v|D_v)$,
3. for all $v \in V : P(v|D_v) > 0$, and
4. for all $v \in V : P(v|D_v)$ is known precisely.

The goal here is to transform G_d to G_u so that G_u is decomposable. By accomplishing this goal, a non–iterative algorithm for calculating the maximum entropy distribution is obtained by applying equation (3.9) to G_u. Spiegelhalter uses the following result in transforming a directed graph to an undirected graph.

Theorem 2 *Suppose a causal graph G_d has no triples of nodes x, y, z such that $x \rightarrow y, y \rightarrow z$ but neither $x \rightarrow y$ nor $y \rightarrow x$. Then G_u is triangulated, and the joint distribution is decomposable.*

Proof: Theorem 2 in Spiegelhalter's paper [31]. ■

We now examine Spiegelhalter's technique for estimating the probability distribution.

1. He assumes the input is a directed graph G_d and all conditional probabilities of the form $P(v|D_v)$ for all v.

2. He changes the directed graph G_d to G_d^* where G_d^* does not have any two unconnected nodes with a common child. (This step is required so theorem 2 will apply.) He does this by selectively adding an edge between a pair of unconnected nodes with a common child. To perform this step, first he labels any node with no descendants as n, then he labels in inverse order the next node by the rule: among those which have all their children labeled, choose the one which has a child with the lowest label, breaking ties at random. Then he uses the algorithm given by Tarjan and Yannakakis [34] to provide a "fill-in" for the ordering of the nodes given by these labels, that will make the graph triangulated.

3. He transforms the directed graph G_d^* to the corresponding undirected graph G_u by removing the directions on the links.

4. As the model choose the set of maximal cliques, $C = \{E_1, \ldots, E_k\}$. Although for arbitrary graphs the problem of finding the maximal cliques is NP-complete, there are efficient algorithms for finding the maximal cliques of triangulated graphs. (See page 268 of Rose, Tarjan and Lueker's paper [29]).

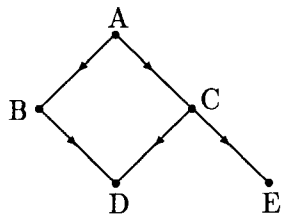


Figure 3.7: The original causal graph G_u.

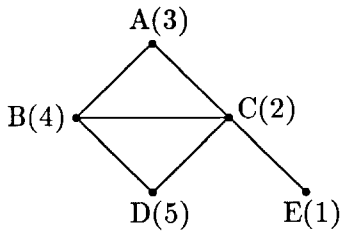


Figure 3.8: The undirected graph, G_u, corresponding to G_d. From the ordering of the vertices which are shown in parenthesis, we can find an ordering of the cliques $\{ABC, BCD, CE\}$ which obeys the running intersection property. The ordering is $E_1 = \{CE\}$, $E_2 = \{ABC\}$, $E_3 = \{BCD\}$.

5. Finally, he applies equation (3.9) to C to obtain a formula for the estimated probability distribution.

We shall work through his technique on an example. Assume we begin the the causal graph G_d which is shown in figure 3.7. In this graph, node D has parents B and C where B and C are not connected. Thus, an edge must be added between them. By theorem 2, the corresponding undirected graph, G_u, is triangulated (see figure 3.8). Since the maximal cliques in G_u are $C = \{ABC, BCD, CE\}$, the joint marginal distributions $P(ABC)$, $P(BCD)$, and $P(CE)$ must be calculated and stored. (Recall that $P(E_i)$ represents a table of $2^{|E_i|}$ joint probabilities.) In order to apply equation (3.9), we must find an ordering of the cliques which obey the running inter-

section property. To find such an ordering, first assign any vertex (we have arbitrarily chosen E) the number 1. Then perform the maximum cardinality search of Tarjan and Yannakakis [34] to order the remaining vertices. (See figure 3.8). Finally order the cliques according to the largest number assigned to any vertex in the clique. By ordering the cliques in this manner, they will have the running intersection property. Once the cliques are ordered we can construct the following table:

r	E_r	V_r	c_r
1	CE	CE	-
2	ABC	ABCE	C
3	BCD	ABCDE	BC

Applying equation (3.9) gives the following formula for the maximum entropy distribution:

$$P^*(V) = P(CE)\frac{P(ABC)}{P(C)}\frac{P(BCD)}{P(BC)}. \tag{3.10}$$

A major difference between his work and our work is that he assumes that the data is given as *conditional* probability distributions rather that joint marginal distributions. From these conditional probabilities he finds a set of joint marginal distributions which form a decomposable model. To obtain the needed data, he used equation (2.5) to calculate the joint probability distributions. Since equation (2.5) only gives the maximum entropy distributions for networks which are singly–connected, in general Spiegelhalter's technique does not produce the maximum entropy distributions.

Chapter 4

Comparisons

In this chapter, we compare our new technique for calculating the maximum entropy probability distribution to Cheeseman's summation technique, Spiegelhalter's algorithm, and Kim and Pearl's algorithm. In section 4.1, we show that our algorithm is at least as efficient as the iterative algorithm with Cheeseman's summation technique applied to it. In section 4.2, we compare our algorithm and Spiegelhalter's algorithm. And finally in section 4.3, we compare our work to Kim and Pearl's work on singly–connected networks.

4.1 Cheeseman's Summation Technique

We begin by comparing our algorithm to Cheeseman's algorithm. We prove that for a given problem, the hyperedges (tables) added by our technique are like the intermediate tables used by Cheeseman's summation technique. The only difference between these tables is that for Cheeseman's technique they are half the size, since they are summed over one of the variables in the table.

As we mentioned in section 2.1, the order chosen for eliminating the variables in Cheeseman's summation technique greatly affects the success of this technique. We want to determine the minimum set of intermediate tables that must be created for a given ordering of the variables. When

evaluating a sum one can visualize an imaginary "scan line" moving from left to right across the hypergraph, where all variables to the left of the scan line have already been summed over. According to the position of a scan line the vertices of the graph may be divided into three parts ("eliminated", "boundary", and "unseen") as follows:

1. $V_E = \{v \in V \mid v \text{ is left of the scan line } \}$
2. $V_B = \{v' \in V - V_E \mid \exists v \in V_E : (v, v') \in E \}$
3. $V_U = V - (V_E \cup V_B)$

Let $G_1, G_2, \ldots, G_l$ be the connected components of the subgraph induced by $V_E \cup V_B$. And let $\gamma(G_i)$ denote the set of vertices of V_E in G_i, and $\Gamma(G_i)$ denote the set of vertices in V_B adjacent to some vertex in G_i (i.e. $\Gamma(G_i)$ is the "neighborhood" of G_i). Finally, $\Gamma'(G_i)$ denotes the subset of $\Gamma(G_i)$ consisting of the vertices adjacent to some vertex in V_U or V_B. Next, we partition the edges of the graph as follows:

1. $E_L = \{(v_i, v_j) \mid v_i \in V_E \text{ and } v_j \in V_E\}$
2. $E_M = \{(v_i, v_j) \mid v_i \in V_E \text{ and } v_j \in V_B\}$
3. $E_R = \{(v_i, v_j) \mid v_i \in V_B \cup V_U \text{ and } v_j \in V_B \cup V_U\}$

Theorem 3 *Given an ordering of the vertices in G, the intermediate tables required by Cheeseman's summation technique are a collection of subsets of the vertices $X_1, X_2, \ldots, X_l$ such that*

$$(\forall i)(\exists j) \mid \Gamma'(G_i) \subseteq X_j. \tag{4.1}$$

Proof: Initially we have the summation

$$\alpha \sum_V \alpha_1 \cdots \alpha_m. \tag{4.2}$$

Since the edges of the graph are partitioned into E_R, E_M, and E_L, and edges in E_R do not contain any vertices in V_E, equation (4.2) can be rewritten as

$$\sum_{V_U \cup V_B} \{\alpha_{E_R}\} \sum_{V_E} \{\alpha_{E_M}\}\{\alpha_{E_L}\} \tag{4.3}$$

where $\{\alpha_{E_R}\}$ represents the product of all the α's corresponding to edges in E_R, and likewise for $\{\alpha_{E_M}\}$ and $\{\alpha_{E_L}\}$. Next, we partition the vertices in V_E according to the connected components $G_1, \ldots, G_l$. The subgraphs are partitioned so that each vertex in V_B must only be connected to elements in V_E which are in the same connected component. Thus, (4.3) becomes

$$\sum_{V_U \cup V_B} \{\alpha_{E_R}\} \sum_{\gamma(G_1)} \{\alpha_{E_L}\}\{\alpha_{E_M}\} \cdots \sum_{\gamma(G_l)} \{\alpha_{E_L}\}\{\alpha_{E_M}\}. \tag{4.4}$$

Any vertex in $\Gamma(G_i) - \Gamma'(G_i)$ is not an element of E_R. So, (4.4) can be written as:

$$\sum_{V_U \cup V_{B'}} \{\alpha_{E_R}\} \sum_{\gamma'(G_1)} \{\alpha_{E_L}\}\{\alpha_{E_M}\} \cdots \sum_{\gamma'(G_l)} \{\alpha_{E_L}\}\{\alpha_{E_M}\} \tag{4.5}$$

where $V_{B'} = \Gamma'(G_1) \cup \cdots \cup \Gamma'(G_l)$ and $\gamma'(G_i) = \gamma(G_i) \cup \Gamma(G_i) - \Gamma'(G_i)$.

We want to replace the summations over the partitions $\gamma'(G_1), \ldots, \gamma'(G_l)$ by a table containing all values needed for $\sum_{V_U \cup V_{B'}}$. The table need not include the α's for the edges in E_L since both vertices are in $\gamma(G_i)$. Similarly, the table need not include the α's for edges in E_M where $v \in V_B \in \Gamma(G_i) - \Gamma'(G_i)$ since both vertices are in $\gamma'(G_i)$. However, since the α's for edges in E_M where $v \in V_B \in \Gamma'(G_i)$ will not be eliminated when evaluating $\sum_{\gamma'(G_i)}$, they must be included in the tables. So clearly $X_1, \ldots X_l$ is sufficient for Cheeseman's algorithm if $(\forall i)(\exists j) \mid \Gamma'(G_i) \subseteq X_j$. That is, for each partition a table will be needed covering all possible values of $\Gamma'(G_i)$.

Finally, we will prove that any set $X_1, \ldots, X_l$ usable in Cheeseman's algorithm satisfies (4.1). Assume otherwise. That is $(\exists i)(\forall j) \mid \Gamma'(G_i) \not\subseteq X_j$. Let $\Gamma'(G_*)$ be the component which is not contained in any table.

In equation(4.5) the $\sum_{V_U \cup V_{B'}}$ goes over all possible values of $\Gamma'(G_*)$. By definition of $\Gamma'(G_*)$ there must be at least one edge in E_R where an element of $\Gamma'(G_*)$ is an endpoint.Therefore, a partial value from $\sum_{\gamma'(G_*)}$ will be needed for $\sum_{V_U \cup V_{B'}}$. This gives us the desired contradiction. ■

Our algorithm requires building the additional tables which correspond to the constraints added to make the given hypergraph acyclic. Just as we did above, we would like to know what additional tables are required given an ordering of the vertices. We will use the following notation.

- Let $v_1, \ldots, v_n$ be an ordering of the vertices.
- Let $H_i = \begin{cases} H & \text{if } i = 0 \\ H_{i-1} + \phi(v_i) - \{e \mid v_i \in e\} & \text{otherwise} \end{cases}$
- Let $\phi(v_i) = \{v \mid v \text{ is adjacent to } v_i \text{ in } H_{i-1}\}$.
- Let $\Phi(v_i) = \begin{cases} \emptyset & \text{if } \phi(v_i) = \emptyset \\ \phi(v_i) \cup v_i & \text{otherwise} \end{cases}$

Theorem 4 *Given an ordering of the vertices of a hypergraph H, the set of all non-empty $\Phi(v_i)$ when added to H will make H acyclic.*

Proof: We begin by showing that in order to eliminate v_i it is necessary to add the hyperedge, e' containing v_i and all vertices adjacent to v_i. That is, we must add $e' = v_i \cup \phi(v_i) = \Phi(v_i)$ to eliminate v_i. By adding e', all edges in H containing v_i will be subsumed by e', and eliminated by reduction step 2 of Graham's algorithm. Since v_i only appears in e', it is removed by reduction step 1, leaving the hyperedge, $e'' = \phi(v_i)$. So, when going from H_{i-1}(the graph after eliminating v_{i-1}) to H_i(the graph after eliminating v_i), the edge set $\phi(v_i)$ must be added and $\{e \mid v_i \in e\}$ must be deleted.

Now we will show that e' must be added in order to eliminate v_i. If e' (or any edge containing e') was not added then at least one of the edges of H containing v_i will remain after applying reduction step 1. This means

when applying reduction step 2 both this edge and e' will contain v_i, so v_i cannot be eliminated. ■

Note that sometimes a vertex may be eliminated during a step in which the goal is to eliminate a different vertex. So if $\phi(v_i)$ is empty no additional hyperedge is needed to eliminate v_i.

Now we are ready to prove that the additional tables required by our algorithm are equivalent to the intermediate tables used in Cheeseman's technique except that Cheeseman's tables each contain one less variable.

Theorem 5 *Given the same graph (hypergraph) and ordering of the vertices in the graph, the intermediate tables $X_1, \ldots, X_j$ used by Cheeseman's summation technique are equivalent to the non-empty $\{\phi(v_i)\}$ where the corresponding set $\{\Phi(v_i)\}$ are the hyperedges added by our technique.*

Proof: Let v_i be the vertex which is being eliminated next in our problem. Then V_E in the Cheeseman's summation technique corresponds to $v_1, \ldots, v_{i-1}$ (i.e. the vertices which have already been eliminated). When v_i is eliminated a hyperedge, e', containing those vertices in $\phi(v_i)$ is added. Thus, the vertices in $\phi(v_i)$ form a clique after v_i is eliminated. Let the vertices be divided into equivalence classes where any two vertices in the same edge (i.e. adjacent vertices) are in the same equivalence class. Clearly, when v_i is eliminated all vertices in $\phi(v_i)$ or adjacent to some vertex in $\phi(v_i)$ are in the same equivalence class. These equivalence classes are identical to the equivalence classes given by $\Gamma(G_1), \ldots, \Gamma(G_l)$ for Cheeseman's summation technique. Finally any vertex in $\Gamma(G_i) - \Gamma'(G_i)$ corresponds to a vertex which was only connected to vertices in V_E (which have been eliminated). That is, the vertices in $\Gamma(G_i) - \Gamma'(G_i)$ are eliminated "for free", so they correspond to $\phi(v_i)$ which are empty. Therefore all non-empty $\phi(v_i)$ are equivalent to $X_1, \ldots, X_l$ where $(\forall i)(\exists j) \mid \Gamma'(G_i) \subseteq X_j$. ■

To better understand theorem 5, we shall look at the example shown in figure 4.1. To explain this example we introduce new notation, where

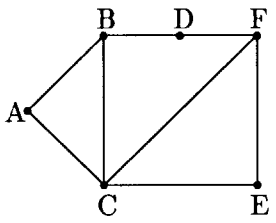


Figure 4.1: Example used to compare our technique to Cheeseman's technique.

the variable sets for the intermediate tables are put in parenthesis above the summation sign. Consider the vertex ordering $DEFCAB$; given such a vertex ordering, theorem 3 specifies the temporary tables needed by Cheeseman's summation technique.

$$P^*(\emptyset) = \sum_{AB} \alpha_{AB} \overset{(AB)}{\sum_{C}} \alpha_{AC}\alpha_{BC} \overset{(BC)}{\sum_{F}} \alpha_{CF} \overset{(BF,CF)}{\sum_{E}} \alpha_{CE}\alpha_{EF} \overset{(BF)}{\sum_{D}} \alpha_{BD}\alpha_{DF}$$

(This summation is only being used for explanatory purposes. Since the elements of the probability distribution sum to 1, we know that $P^*(\emptyset) = 1$.) When evaluating $\sum_D$ it will be necessary to keep a table with the value of $\alpha_{BD}\alpha_{DF}$ for Ω_{BF}. When evaluating $\sum_E$ as well as keeping the table for B and F, another table with all combinations of C and F must be created. Below are the temporary tables used by Cheeseman's method.

$$BF, CF, BC, AB$$

Now we consider the additional hyperedges which must be added to make the hypergraph acyclic. Theorem 4 defines the set of additional hyperedges that make a hypergraph acyclic. First, the hyperedge BFD eliminates vertex D. Hyperedge BFD eliminates hyperedges BD and DF since it subsumes them. Now D is only in hyperedge BDF and so is eliminated, leaving hyperedge BF. Second, hyperedge CFE eliminates vertex E. Now

vertex F is only in hyperedges BF and CF so BCF eliminates it. Finally, hyperedge ABC eliminates the remaining vertices. So, the following additional hyperedges will make our example graph acyclic (in parenthesis is the variable eliminated by adding the edge):

$$BF(D), CF(E), BC(F), AB(C)$$

Ignoring the variables in parentheses, these are identical to Cheeseman's tables. Nevertheless, there are important differences between these methods.

First, in terms of time complexity, Cheeseman's method specifies an iterative approximation of the αs, whereas our method requires no such iteration. So, if Cheeseman's method requires 10 iterations on the average, our method should yield an average speed-up of a factor of 10.

Second, in terms of space complexity, both methods use approximately the same amount of space. However, our method adds what might be called "permanent" edges, since they correspond to tabulations of the raw data. Note, however, that new edges may subsume and eliminate original edges, so the space required by our method may not be quite as great as it first appears. In Cheeseman' method the tables exist only temporarily during the course of the computation, and not all such tables may be needed at the same time.

And finally, in terms of the "precomputation" needed, both methods need to compute a vertex ordering to use. We observe that a good summation ordering is a good ordering for eliminating vertices. So the problem of choosing the hyperedges to make a graph acyclic is essentially equivalent to the problem of choosing an optimal summation ordering. Therefore, we conclude that our algorithm will be generally more efficient than Cheeseman's algorithm where both are applicable.

4.2 Spiegelhalter's Algorithm

We now compare our algorithm to Spiegelhalter's algorithm. We have shown that a graphical model is decomposable if and only if the corresponding hypergraph is acyclic. We will prove that if the decomposable model obtained by Spiegelhalter's technique is the same as the acyclic hypergraph obtained after step 4 of our technique, then these techniques produce the same formula for the estimated probability distribution.

We begin by proving that a graphical model is decomposable exactly when it is acyclic.

Theorem 6 *A graphical model is decomposable if and only if the corresponding hypergraph is acyclic* [1].

Proof: Given a set of attributes and constraints, construct a graph G with a vertex for each attribute and edge connecting the vertices corresponding to each pair of attributes contained in the same constraint. Likewise, construct a hypergraph H with a vertex for each attribute and a hyperedge corresponding to each constraint. By definition, G is decomposable if it is graphical and triangulated, and a H is acyclic if it is conformal and $C(H)$ is chordal. By definition G is decomposable exactly when H is conformal. We also note that G and $C(H)$ are the same graph, so G is triangulated exactly when $C(H)$ is chordal. Therefore, G is graphical if and only if H is acyclic. ■

The essence of our algorithm is to take an arbitrary hypergraph and transform it into an acyclic hypergraph. Likewise, Spiegelhalter's algorithm takes an arbitrary directed graph and transforms it into a decomposable model. From theorem 6, we know that a model is decomposable if and only if the corresponding hypergraph is acyclic. So the first question one may ask

[1] Malvestuto [25] pointed out that both acyclic hypergraphs and decomposable models have been shown to obey the running intersection property, which implies that these models must be equivalent.

is how does our technique and Spiegelhalter's technique compare given that they arrive at the same acyclic hypergraph. In the next theorem, we prove that the algorithm Spiegelhalter's technique uses once he has a decomposable model produces the same formula for the estimated probability distributions as the algorithm our technique uses once we have an acyclic hypergraph.

Theorem 7 *Given an acyclic hypergraph (decomposable model) as input, equation (3.9) and equation (3.1) give the same formula for the estimated probability distribution.*

Proof: First of all since equation (3.9) applies to decomposable models and equation (3.1) applies to acyclic hypergraphs, by theorem 6 these equations apply to the same graphs. Equation (3.9) can be rewritten as

$$P^*(V) = \left(\prod_{r=1}^{m} P(E_r)\right)\left(\prod_{r=1}^{m} \frac{1}{P(c_r)}\right), \tag{4.6}$$

and equation (3.1) can be rewritten as

$$P^*(V) = \left(\prod_i E_i^{(0)}\right)\left(\prod_{k=0}^{l-1} \frac{\prod_i P\left(E_i^{(k+1)}\right)}{\prod_i P\left(Y_i^{(k)}\right)}\right). \tag{4.7}$$

We know that

$$\prod_{r=1}^{m} P(E_r) = \prod_i E_i^{(0)}$$

since those are just the product of the supplied joint marginal probabilities. So all we must do show that the second parts of equations (4.6) and (4.7) are equivalent.

Order the edges in $\mathcal{E}^{(0)}$ according to the level reached in the reduction procedure so that the earlier an edge is eliminated the higher it is numbered. Now we will show that this ordering will obey the running intersection property. Let $V_t = E_1 \cup \cdots \cup E_t$ where E_1 is the first edge in the ordering, and E_t is the last edge in the ordering. By definition

$$Y_i^{(k)} = \left(\bigcup_{j \neq i} E_j^{(k)}\right) \bigcap E_i^{(k)}. \tag{4.8}$$

We are interested in the value of $Y_i^{(k)}$ at the value for k when edge E_i is eliminated. In this case, the term $\cup_{j \neq i} E_j^{(k)}$ will consist of the union of what remains after step k of all the edges ordered before E_i. That is, $\cup_{j \neq i} E_j^{(k)}$ contains all vertices in edges in V_{i-1} except for those which have been eliminated because they were only in one hyperedge. Since these eliminated vertices could not be in $E_i^{(k)}$, it follows that for the step k in which $Y_i^{(k)}$ is eliminated

$$Y_i^{(k)} = V_{i-1} \cap E_i^{(k)} = V_{i-1} \cap E_i^{(0)} = V_{i-1} \cap E_i. \tag{4.9}$$

Furthermore, from the ordering of the hyperedges, we know that each hyperedge is subsumed by a predecessor in the ordering. So, $Y_i^{(k)} \subseteq E_j$ for $i > j$. Thus, from equation (4.9) is follows that

$$E_i \cap V_{i-1} = c_i = Y_i^{(k)} \subseteq E_j \text{ where } i > j \tag{4.10}$$

which proves that the running intersection property holds.

Now let's look at

$$\left(\prod_{k=0}^{l-1} \frac{\prod_i P\left(E_i^{(k+1)}\right)}{\prod_i P\left(Y_i^{(k)}\right)} \right).$$

For any k in which $Y_i^{(k)}$ is not eliminated, $E_i^{(k+1)} = Y_i^{(k)}$. Therefore all terms will cancel except those for which $Y_i^{(k)}$ is eliminated in step k. We have argued above that in this case, $Y_i^{(k)} = E_i \cap V_{i-1} = c_i$. So we have that

$$\left(\prod_{k=0}^{l-1} \frac{\prod_i P\left(E_i^{(k+1)}\right)}{\prod_i P\left(Y_i^{(k)}\right)} \right) = \prod_{r=1}^{m} \frac{1}{P(c_r)} \tag{4.11}$$

And thus equations (4.6) and (4.7) are the same. ■

So we know that if our technique and Spiegelhalter's technique obtain the same acyclic hypergraph for a problem, then they will arrive at the same formula for the estimated probability distribution.

We now consider the additional data required by our technique and the additional data required by Spiegelhalter's technique. As we have mentioned, adding hyperedges to the original hypergraph corresponds to requiring joint marginal probability distributions which were not included in the original data. Now let's look at the data requirements for Spiegelhalter's algorithm. In order to get a decomposable model, Spiegelhalter must add edges to the original directed graph. In doing so, he may form cliques in the corresponding undirected graph which are larger than any of the original maximal cliques. These large cliques correspond to joint marginal probability distributions which are not contained in the original data. In fact, these two techniques will require the same additional data exactly when the two techniques obtain the same acyclic hypergraph.

Finally, let's compare how we propose getting the additional data to how Spiegelhalter proposes doing so. We propose that this additional data is collected with the original data by having the first four steps of our algorithm be "planning" steps. On the other hand, Spiegelhalter's technique assumes conditional independence to get the additional data from the given constraints. That is, he proposes to obtain these joint marginal distributions by applying equation (2.5) to the directed graph given as input. Now in the case where the directed graph is a tree (singly–connected) then the joint marginal distribution obtained will be the maximum entropy distribution. However, when the directed graph is not a tree, the resulting joint marginal distribution will not be the maximum entropy distribution. Therefore, even in the case where our techniques arrive that the same formula for the estimated probability distribution, the actual estimates for the complete probability distribution may differ since the data for the added joint marginal distributions may be different.

4.3 Kim and Pearl's Work

In this section we will compare acyclic hypergraphs to singly–connected Bayesian networks. Kim and Pearl's technique for propagating new evidence through the network depends on the network being singly–connected [2]. A Bayesian network is a directed acyclic graph; such a network is said to be *singly-connected* if it has no *undirected* cycles.

Given a network N, we define a corresponding hypergraph G as follows. The vertices of G are the nodes of N. For each node x in N we create a hyperedge in G, consisting of the corresponding vertex and the vertices corresponding to all immediate predecessor of x in N.

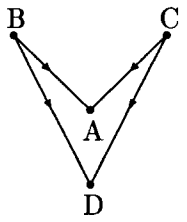
Theorem 8 *If a Bayesian network N is singly-connected, then the corresponding hypergraph G is acyclic.*

Proof: Since N is singly–connected, it contains no undirected cycles, by the definition of singly-connected. Since an acyclic undirected graph is a tree or forest, N must contain some node s with degree at most one. The corresponding vertex in G is contained in exactly one hyperedge and so can be eliminated by reduction step 1. Finally, we must show that the reduced network N' so obtained corresponds to the reduced hypergraph G' that remains after eliminating the vertex corresponding to s. When s is a source in N (or a sink which is the second to last node) this correspondence is obtained immediately. In the remaining cases, the correspondence holds only after applying reduction step 2 to eliminate the hyperedge remaining after s is eliminated. (This hyperedge contains only the parent of s.) The network N' is singly–connected. By induction, every node in G can be eliminated. Thus, G is acyclic. ■

Theorem 9 *A Bayesian network is not necessarily singly-connected if the corresponding hypergraph is acyclic.*

[2] Pearl [26] has examined ways to extend his algorithm to multiply connected networks.

Proof: We prove this by means of an example. The following Bayesian network is not singly-connected since there is an undirected cycle.



The hypergraph corresponding to the above network is acyclic as shown by the reduction below:

$$\{A\underline{BC}, \underline{BC}D\}$$
$$\Downarrow$$
$$\{BC\}$$
$$\Downarrow$$
$$\emptyset$$

■

We know that in the case where the Bayesian network is a tree, equation (2.5) gives the maximum entropy distribution, and when the network is not a tree, equation (2.5) does not give the maximum entropy distribution. However, if the independence constraints assumed by Kim and Pearl are supplied as additional constraints to a maximum entropy algorithm, then the two techniques give equivalent results. This is because the conditional independence constraints uniquely define a probability distribution. So, in some sense one could argue that a maximum entropy algorithm is more general that the one used by Kim and Pearl.

Chapter 5

Conclusions and Open Problems

We have presented an efficient algorithm for calculating the maximum entropy distribution for a given set of attributes and constraints. Using a hypergraph to model the attributes and constraints, we have shown the benefits of making the corresponding hypergraph *acyclic*. We also have shown how to make a hypergraph acyclic by adding hyperedges (constraints). We have proved that our technique is at least as efficient as Cheeseman's method. Furthermore, we proved that acyclic hypergraphs and decomposable models are equivalent properties. Also we demonstrated that the formula for the maximum entropy distribution which is derived for acyclic hypergraphs by our technique and the formula given for decomposable models by Spiegelhalter's technique are equivalent if given the same undirected graph as input. The significant difference between these techniques is that we "plan ahead" so that the additional data required is available. Finally we have compared our work to Kim and Pearl's work on singly–connected networks.

An open problem is to determine whether or not the problem of choosing the best set of hyperedges which will make a hypergraph acyclic is an NP–complete problem. We conjecture that this is the case, but have not yet been able to exhibit a proof. We would like either to find an NP–completeness

proof, or to find a polynomial time algorithm to solve the problem. Given our conjecture that the *constraint addition* problem is NP–complete, we would like to explore heuristics for finding a good set of hyperedges to add.

Another direction of future research is to determine how well our new algorithm works on real–life problems. We intend to try our technique on some realistic examples. Our goal is to determine if the size of the hyperedges will remain within reasonable limits for such realistic examples. We expect that in practice our new method will give substantial improvements in running time. Finally, we will study the effects on accuracy of keeping tables which may be larger than the original tables.

Another interesting problem is to find a condition which ensures the existence of a non–iterative algorithm for approximating the maximum entropy distribution when the input can consist of both joint probabilities and conditional probabilities, as well as some clearly defined independence constraints. Furthermore, since the problem of finding the maximum entropy distribution is a non–linear optimization problem, it is interesting to ask if the techniques explored in this thesis could be applied to the general problem of non–linear optimization.

Bibliography

[1] Agmon, N., Y. Alhassid, R.D. Levine, "An Algorithm for Determining the Lagrange Parameters in the Maximal Entropy Formalism," In Levine and Tribune, editors, *The Maximum Entropy Formalism*, M.I.T. Press, (1979).

[2] Agmon, N., Y. Alhassid, R.D. Levine, "An Algorithm for Finding the Distribution of Maximal Entropy," *Journal of Computational Physics* **30**,2 (February 1979), 250–259.

[3] Beeri, C., R. Fagin, D. Maier, A. Mendelzon, J.D. Ullman and M. Yannakakis, "Properties of Acyclic Database Schemas," in *Proc.* 13^{th} *Annual ACM STOC* (1981), 355–362.

[4] Beeri, C., R. Fagin, D. Maier and M. Yannakakis, "On the Desirability of Acyclic Database Schemas,"*J. ACM*, **30**,3 (1983), 355-362.

[5] Brown, D.T., "A Note on Approximations to Discrete Probability Distributions," *Information and Control*, **2** (1959), 386–392.

[6] Cheeseman, P.C., "A Method For Computing Generalized Bayesian Probability Values For Expert Systems," in *Proc. Eighth International Conference on Artificial Intelligence* (August 1983), 198–202.

[7] Cheeseman, P.C., "Learning of Expert Systems From Data," in *Proc. IEEE Workshop on Principles of Knowledge Based Systems* (1984), 115–122.

[8] Chow, C.K. and C.N. Liu, "Approximating Discrete Probability Distributions With Dependence Trees,"*IEEE Trans. on Info. Theory*, **IT-14**,3 (May 1968), 462–467.

[9] Csiszár, I., "I-Divergence geometry of probability distributions and minimization problems," *Annals of Probability*, **3**,1 (1975), 146–158.

[10] Dalkey, N.C., "Min-Score Inference on Probability Systems," *University Of California, Los Angeles Dept. of Computer Science Technical Report UCLA-ENG-CSL-8112*, (June 1981).

[11] Darroch, J.N., S.L. Lauritzen, and T.P. Speed, "Markov Fields and Log–Linear Models for Contingency Tables," *Annals of Statistics*, **8** (1980), 522–539.

[12] Edwards, D., and S. Kreiner, "Analysis of contingency tables by graphical models," *Biometrika* **70**,3 (1983), 553–565.

[13] Fienberg, S.E., "An Iterative Procedure For Estimation In Contingency Tables," *The Annals of Mathematical Statistics*, **41**,3 (1970), 907–917.

[14] Geman, S., "Stochastic Relaxation Methods For Image Restoration and Expert Systems," In Cooper, D.B., R.L. Launer, and E. McClure, editors, *Automated Image Analysis: Theory and Experiments*, New York: Academic Press, (to appear).

[15] Graham, M.H., "On the Universal Relation," *University of Toronto Technical Report* (1979).

[16] Ireland, C.T., and S. Kullback, "Contingency tables with given marginals," *Biometrika* **55**,1 (1968), 179–188.

[17] Jaynes, E.T., "Where Do We Stand On Maximum Entropy," In Levine and Tribune, editors, *The Maximum Entropy Formalism*, M.I.T. Press, (1979).

[18] Jaynes, E.T., "On the Rationale of Maximum-Entropy Methods," *Proceedings of the IEEE*, **70**,9 (September 1982), 939–952.

[19] Johnson, R.W., and J.E. Shore, "Comments and corrections to 'Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy'," *IEEE Trans. Inform. Theory* **IT-29**, 6 (Nov. 1983), 942–943.

[20] Ku, H.H. and S. Kullback, "Approximating Discrete Probability Distributions," *IEEE Trans. on Info. Theory*, **IT-15**,4 (July 1969), 444–447.

[21] Kim, J.H. and J. Pearl, "A Computational Model for Causal and Diagnostic Reasoning in Inference Systems," *Proc. Eighth International Conference on Artificial Intelligence* (August 1983), 190–193.

[22] Lewis, P.M., "Approximating Probability Distributions to Reduce Storage Requirements," *Information and Control*, **2** (1959), 214–225.

[23] Lippman, A.F., "A Maximum Entropy Method for Expert System Construction," PhD thesis, Brown University, Division of Applied Mathematics, (May 1986).

[24] Malvestuto, F.M., "Decomposing Complex Contingency Tables to Reduce Storage Requirements," *Proceedings of 3rd International Workshop on Statistical and Scientific Database Management* (July 1986), 66–71.

[25] Malvestuto, F.M., "Modeling Large Bases of Categorical Data with Acyclic Schemes," *Proceedings of the International Conference on Database Theory* (September 1986).

[26] Pearl, J., "Fusion, Propagation and Structuring in Bayesian Networks," *University Of California, Los Angeles Dept. of Computer Science Technical Report CSD-850022 R-42*, (April 1985).

[27] Rissanen, J., "A Universal Prior for Integers and Estimation by Minimum Description Length," *The Annals of Statistics*, **11**,2 (1983), 416–431.

[28] Rose, D.J. and R.E. Tarjan, "Algorithmic Aspects of Vertex Elimination in Directed Graphs," *SIAM Journal Applied Math*, **24** (1978), 176–197.

[29] Rose, D.J., R.E. Tarjan, and G.S. Lueker, "Algorithmic Aspects of Vertex Elimination on Graphs," *SIAM Journal Comput.*, **5**,2 (June 1976), 266–283.

[30] Shore, J.E., and R.W. Johnson, "Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy," *IEEE Trans. Inform. Theory*, **IT-26**,1 (Jan. 1980), 26–37.

[31] Spiegelhalter, D.J., "Probabilistic Reasoning in Predictive Expert Systems," in *Uncertainty in Artificial Intelligence*, (eds. Kanal, L.N. and Lemmer, J.) North Holland Amsterdam, 47–68.

[32] Spiegelhalter, D.J., "Coherent Evidence Propagation in Expert Systems," to appear in the *The Statistician.*

[33] Strang, G., *Introduction to Applied Mathematics,* Wellesley–Cambridge Press: Wellesley, Massachusetts (1986), 367–470.

[34] Tarjan, R.E. and M. Yannakakis, "Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs," *SIAM J. Comp.*, **13**,3 (August 1984), 566–579.

[35] Tikochinsky, Y., N.Z. Tishby, and R.D. Levine, "Consistent inference of probabilities for reproducible experiments," *Physical Rev. Letters* **52**, 16 (16 April 1984), 1357–1360.

[36] Tversky, A. and Kahneman, D., "Judgment Under Uncertainty: Heuristics and Biases," *Science*, **185**, (September 1974), 1124–1131.

[37] Wermuth, N. and S.L. Lauritzen, "Graphical and Recursive Models for Contingency Tables," *Biometrika* **70** (1983), 537–552.

[38] Yannakakis, M., "Computing the Minimum Fill-in is NP-Complete," *SIAM Journal Alg. Disc. Meth.*, **2**,1 (March 1981), 77–79.