Beyond Independence: Probabilistic Models for Query Approximation on Binary Transaction Data

Dmitry Pavlov
NEC Research Institute,
Princeton, NJ, 08540
`dpavlov@research.nj.nec.com`
Heikki Mannila
HIIT Basic Research Unit,
Department of Computer Science
University of Helsinki, P.O. Box 26,
FIN-00014 Helsinki, Finland
`Heikki.Mannila@cs.Helsinki.FI`
Padhraic Smyth
Information and Computer Science,
University of California, Irvine,
Irvine, CA 92697-3425
`smyth@ics.uci.edu`

Abstract

We investigate the problem of generating fast approximate answers to queries posed to large sparse binary data sets. We focus in particular on probabilistic model-based approaches to this problem and develop a number of techniques that are significantly more accurate than a baseline independence model. In particular, we introduce two techniques for building probabilistic models from frequent itemsets: the itemset maximum entropy method, and the itemset inclusion-exclusion model. In the maximum entropy method we treat itemsets as constraints on the distribution of the query variables and use the maximum entropy principle to build a joint probability model for the query attributes online. In the inclusion-exclusion model itemsets and their frequencies are stored in a data structure called an ADtree that supports an efficient implementation of the inclusion-exclusion principle in order to answer the query. We empirically compare these two itemset-based models to direct querying of the original data, querying of samples of the original data, as well as other probabilistic models such as the independence model, the Chow-Liu tree model, and the Bernoulli mixture model. These models are able to handle high-dimensionality (hundreds or thousands of attributes), whereas most other work on this topic has focused on relatively low-dimensional OLAP problems. Experimental results on both simulated and real-world transaction data sets illustrate various fundamental tradeoffs between approximation error, model complexity, and the online time required to compute a query answer.

Index Terms

Binary transaction data, query approximation, probabilistic model, itemsets, ADTree, maximum entropy.

I. INTRODUCTION

Massive data sets containing huge numbers of records are of increasing interest to organizations that routinely collect such data and to data miners who try to find regularities in them. One class of such data is transaction data with rows corresponding to transactions and columns corresponding to particular items or attributes. This class is typically characterized by *sparseness*, i.e., there may be hundreds or thousands of binary attributes but a particular record may only have a few of them set to 1.

An example of a binary transaction data set is a Web log that records page requests for a particular Web site. The rows (records) in such a data set correspond to various users accessing the site and columns (attributes) correspond to different pages within the site. Clearly, most of the users access only a small fraction of pages, making the data set sparse. Within a single day popular Web sites can produce millions of records.

Query selectivity estimation for such binary data can be defined as follows. Let $R = \{A_1, \ldots, A_k\}$ be a table header with k 0/1 valued attributes (variables) and r be a table of n rows over header R. We assume that $k \ll n$, and that the data are sparse i.e., the average number of 1's per row is substantially smaller than the number of attributes. By definition, a row of the table r satisfies a conjunctive query Q if and only if the corresponding attributes in the query and in the row have equal values. We are interested in finding the number of rows in the table r satisfying a given conjunctive query Q defined on a subset of its attributes.

Query selectivity estimation is an important practical problem in the optimization of database management systems where query profilers and query optimizers routinely rely on fast and reasonably accurate estimates of the query counts [30]. Query selectivity can also be important in interactive data mining applications and exploratory data analysis, where a data miner may be exploring various hypotheses in the data and would prefer to get an approximate count to a query in real-time rather than waiting for an exact count to be generated. From a probabilistic viewpoint we can pose the problem as that of estimating the true frequency of Q in the table r using an approximate probability model P_M.

Any Boolean query can be answered exactly using a single scan through the data set. While this approach has linear complexity and works well for small data sets, it can be prohibitively slow for real-time queries on massive data sets. Consequently, we would like to have approximate algorithms that allow us to trade accuracy in the estimate $P_M(Q)$ with the time and memory taken to calculate it.

There are two main aspects of this paper:

1) We introduce and analyze two probabilistic modelling techniques that are based on frequent itemsets ([2], [3]): (1) the itemset maximum entropy method, and (2) the itemset inclusion-exclusion model. In the maximum entropy method we treat itemsets as constraints on the distribution of the query variables and use the *maximum entropy* principle to build a joint probability model for the query attributes online. In the inclusion-exclusion model we store itemsets and their frequencies in an ADTree data structure. The virtue of the ADtree lies in its ability to efficiently reconstruct the query count based on the inclusion-exclusion principle.

2) We perform a detailed experimental evaluation of various probabilistic models for query approximation, including the maximum entropy method, the itemset inclusion-exclusion model, the independence model as a baseline, the Chow-Liu tree model, and the Bernoulli mixture model. Our experimental study allows us to draw a variety of conclusions on the general characteristics of each approach in terms of memory requirements, the online time taken to answer a query, and the accuracy of the resulting estimate.

Previous work on this problem has investigated the use of wavelets (see, e.g. [20], [33], [7]), multidimensional histograms ([23], [30]) and sampling ([16], [1]) for query approximation. The use of inclusion-exclusion for query approximation has been previously mentioned in [18]. Probabilistic models of various forms have also been investigated in limited contexts. Mixtures of Gaussian independence models were investigated for generating approximate queries on real-valued data sets in [32]. Bayesian networks for the task of selectivity estimation over multiple tables in a relational database were considered in [14]. The use of statistical interaction models in dependency based histograms for query selectivity estimation is described in [12]. However, none of this work contains a complete and a systematic investigation of different *probabilistic* modelling techniques for query approximation. In addition, in all of this prior work the techniques used were only tested on relatively low-dimensional data cubes (10 or fewer dimensions).

In our earlier work on this topic we have reported initial results on the use of maximum entropy methods [17], [24], mixture models, and Bayesian networks [26] for online query answering in sparse binary data sets of high-dimension (50 dimensions and higher). This present paper studies a new query-answering method based on the inclusion-exclusion principle and provides a systematic characterization and empirical evaluation of the performance of a broad set of probabilistic methods for approximate query answering.

The novelty of the work described in this paper can be summarized as follows:

1) The proposed methods work on high-dimensional binary transaction data rather than being limited to relatively low-dimensional aggregate data cubes;

2) We explore a variety of probabilistic models (rather than just the independence model or mixtures). Most of these are not new models (with the exception of the itemset-based maximum entropy and inclusion-exclusion models), but the application of these models to query answering is new; and,

3) We provide extensive theoretical and empirical characterizations of the trade-offs among the different modelling approaches.

The rest of this paper is organized as follows. In Section II we introduce some general notation and discuss the metrics used to compare different models. Section III defines itemsets and discusses their main properties. Section IV provides definitions of all the probabilistic models that we investigate for the query selectivity estimation problem and gives a summary of their worst-case time and memory requirements. Section V presents empirical results and in Section VI we discuss the main conclusions.

II. Notation and Statistics for Model Comparison

We use the following notation in the remainder of the paper. By P we denote the "true" unknown distribution that generated the data and by x_Q the variables of a conjunctive query Q, so that $P(x_Q)$ is the distribution on the query variables and $P(Q)$ is its value for an instantiation of the variables defined by the query. Note that $P(x_Q)$ is a specific distribution over the subset of variables in Q, *not* over all variables in the table R.

We use P_M to denote an estimate of P from the data (i.e., a probabilistic model), $P_M(x_Q)$ is the estimate of $P(x_Q)$, and $P_M(Q)$ is the estimate of $P(Q)$.

We use the following criteria to compare the performance of different models:

Definition II.1: **Time.** The *offline time cost* T_P is the time taken to learn the model or to collect summary information about the data set. The *online time cost* $t_P(Q)$ is the time taken to answer the query Q online using the model P_M.

The online cost $t_P(Q)$ depends directly on the query. One general strategy is to have a fast and accurate online answer at the expense of potentially much higher offline computational investments.

Definition II.2: **Memory.** *Memory* is the complexity of the learned model P_M (e.g., its minimum description length in bits [31]).
We can expect more complex models to generate more accurate answers on the data used to build the model, but also to be more expensive in terms of online time taken to answer a query.

Definition II.3: **Error.** Let $\pi(Q)$ denote the *query distribution* [1], i.e., the probability that a particular query Q is issued. The *error in answering the query* Q, $e_P(Q)$, is defined as the difference between the query true count $C_t(Q)$ on the test data and the count estimated from the model P_M, i.e. $nP_M(Q)$. The relative expected error with respect to $\pi(Q)$ is then

$$E_\pi[|e_P(Q)|]/E_\pi[C_t(Q)].$$

We use the relative empirical error defined as

$$E_{rel} = \frac{\sum_{j=1}^{N_{Q's}} |e_P(Q_j)|}{\sum_{j=1}^{N_{Q's}} C_t(Q_j)}, \tag{1}$$

where $N_{Q's}$ is the number of random query drawings from $\pi(Q)$ and $C_t(Q_j)$ is the true count of the query Q_j.

III. Definition of Itemsets and Their Properties

Definition III.1: An *itemset* associated with the binary table r (with header R) is defined to be either a single positively initialized attribute or a conjunction of mutually exclusive positively initialized attributes from R [2]. We call an itemset *T-frequent* if its count in the table r is at least T, where T is some predefined non-negative threshold. The *size* of an itemset is the number of conjuncts it is defined on.

Itemsets have several attractive properties for binary transaction data. To provide an error-free answer to any query expressed as a Boolean expression on the attributes of the binary table r it is sufficient to know all 0-frequent itemsets and their counts in r. The proof of this statement follows directly from the inclusion-exclusion principle and the definition of the itemsets. We discuss this method in the next section in more detail where we introduce the itemset inclusion-exclusion model and the ADTree data structure (that efficiently implements the inclusion-exclusion principle). In what follows we will use the following two theorems:

Theorem 1: All T-frequent itemsets have a hierarchical structure, i.e., if an itemset on a set of variables X is T-frequent then any itemset on a subset of variables from X is T-frequent as well.

Proof: The proof of this assertion follows immediately from the definition of a T-frequent itemset by contradiction. ∎

Theorem 2: If an itemset on a set of variables X is T-frequent then any count query Q (expressed as an arbitrary Boolean expression) involving variables that are all contained in the set X can be answered exactly.

Proof: Notice first that if query Q mentions variables in the set Y, and only these variables, and $Y \subset X$, then any itemset on variables from Y is T-frequent due to theorem 1. An arbitrary Boolean query Q can be converted to disjunctive normal form (DNF), the probability of which equals the sum of the probabilities of the individual disjuncts due to their mutual exclusiveness.

---

[1] We assume that this distribution is known, but in principle we could learn $\pi(Q)$ for a population of individuals.

Each disjunct is a conjunction of query attributes or their negations whose probability due to the inclusion-exclusion principle can be expressed as a linear combination of probabilities of conjunctions of positively initialized attributes from Y. The former probabilities in turn can be read directly from the T-frequent itemsets (due to theorem 1). ∎

There exist well-known efficient algorithms to compute all itemsets from large binary tables (e.g., [19], [4]). Provided that the data are sparse, the running times of these algorithms have been found in experimental analyses to be linear in both the size of the table and in the number of frequent itemsets. Thus, computing itemsets does not typically incur a high preprocessing cost on sparse data in practice.

How should one select T for a given data set, especially if the memory available to store itemsets is bounded? Assuming the data set is sparse, by setting T originally to half of the number of records we should expect very few (if any) T-frequent itemsets to be present. We can then repeatedly divide T by two, noting how much memory is taken by the set of itemsets for each value of T. We can stop decreasing T when the extracted itemsets exceed a predefined memory bound. In the results in this paper we used this simple approach to find a reasonable range for T values.

We also augment the itemsets in this paper with the set of frequency counts of all individual attributes (i.e., all itemsets on a single attribute are included, whether frequent or not). This simple idea improves the quality of the model in that the marginal probabilities are known to the model for all attributes.

One can in theory also use the information that certain itemsets are *not* T-frequent. For example if we know that the itemset $A = 1 \wedge B = 1$ is not T-frequent, then we know that this joint probability (or count) is constrained to be less than T/n. In theory this information could be used as an inequality constraint (e.g., in the maximum entropy approach described below) in the building of the probabilistic models for queries involving A and B. However, the complexity of doing so would significantly increase the complexity of our proposed methods, so we do not pursue this idea in this paper. It can also be conjectured that since non-frequent itemsets are relatively rare by definition, ignoring information about such itemsets is likely to have a relatively small overall contribution the overall average error rate of a probabilistic model.

From this point forwards the focus of this paper is on how itemsets can be used in probabilistic modelling, and we implicitly assume that itemset generation has already taken place offline.

IV. Probabilistic Models for Query Selectivity Estimation

A. The Training Data Counts (The Linear Scan Method)

Any Boolean query Q can be answered exactly using a single scan through the data set. For each data record we need to establish whether it satisfies query Q or not, keep the count of records that satisfy the query and report the count at the end of the scan. This can be carried out in time that is worst-case linear in both the size of the record and the number of literals in the query Q. This gives a time complexity $O(n_Q \sum_{i=1}^{n} N_{1's}(i))$, where $N_{1's}(i)$ is the size of the i-th record (i.e. the number of 1's in it), n_Q is the number of literals in the query, and n is the number of data records. The memory complexity of the method is worst case linear in both the number of records in the data and the size of the largest data record, $O(\sum_{i=1}^{n} N_{1's}(i))$, which can be a significant cost for massive data sets.

Experimental results reported in Section V illustrate how the time performance of the linear scan method degrades as a function of the number of records. The approximate algorithms discussed in the following subsections allow us to trade the accuracy of the estimate $P_M(Q)$ with the time and memory taken to calculate it.

B. Random Samples

In a manner similar to the linear scan method discussed above in Section IV-A one can randomly sample records from the training data and find the count of the query based on the sample. There are two types of sampling procedures available: (1) offline sampling, where a sample is extracted before any queries are posed to the database, and (2) online sampling, where data records are randomly accessed once the query is posed. In the results reported in this paper we use offline sampling.

Using a sample of size n_S to answer queries reduces the memory and online time requirements by approximately a factor of n/n_S compared to using the linear scan method.

C. The Independence Model

Assuming independence of the attributes leads to the simplest possible probabilistic model—the independence model.

Since the data are binary-valued, we only have to store one parameter per attribute, $\theta_i = P(A_i = 1)$, where the probability is obtained directly from the data as a maximum likelihood estimate, $\theta_i = f_i/n$, where f_i is the number of 1's for attribute i. The probability of a conjunctive query is then approximated by the product of the probabilities of the individual attribute-value combinations occurring in the query, where the i-th factor has the form $\theta^{A_i}(1 - \theta_i)^{(1-A_i)}$ and A_i is the value of the i-th attribute in the query.

The estimation of all θ_i's requires a single scan through the data, during which we just add "1" to the counts corresponding to the positively initialized attributes of a given record. In contrast with scanning the training data it is important to emphasize that the estimation of θ_i's can be performed *offline*, i.e., before the actual query is issued by the user. The estimation incurs linear time complexity in the number of records and in the size of the largest record.

The online time cost is $O(n_Q)$ while the memory is $O(k)$ where k is the number of attributes in the data set, so that both complexities scale well to large data sets. However, as we show in the experimental results in Section V, the quality of the approximations produced by the independence method can be poor. Nonetheless, because of its simplicity, the independence model is widely used in commercial RDBM systems for query selectivity estimation (e.g. [30]).

D. The Chow-Liu Tree Model

The Chow-Liu tree model (also known as the multivariate tree distribution model) is a model that assumes that there are only pairwise dependencies between the variables, and that the dependency graph on the attributes has a tree structure [8]. To fit a distribution using a tree structure it is sufficient to know the pairwise joint probabilities (or marginals) of all the variables. Finding the pairwise marginals can be performed by a single scan through the data set during which we just add "1" to the counts corresponding to all possible pairs of the positively initialized attributes of a given record. We also need to collect the information about the frequencies of the individual attributes in the manner described in the Section IV-C. After a single data scan we would have collected the following statistics $\theta_i = P(A_i = 1)$ and $\theta_{ij} = P(A_i = 1, A_j = 1)$, which are sufficient to reconstruct any pairwise marginal using the inclusion-exclusion principle. The time cost is $O(nM_r^2)$, where M_r is the size of the largest record.

The algorithm that fits the tree consists of two more steps, namely, computing the mutual information between the attributes (complexity is $O(k^2)$) and applying Kruskal's algorithm [9] to find the minimum spanning tree of the full graph whose nodes are the attributes and the weights

on the edges are the mutual information values (complexity is $O(k^2 \log k)$). The overall offline time complexity is upper bounded by either $O(nM_r{}^2)$ or $O(k^2 \log k)$, with the first term likely to dominate for large n such that $n \gg k$. The memory requirements for the algorithm are associated with the storage of the pairwise marginals, and are thus $O(k^2)$.

Once the tree is learned, for a given query Q we first use the chain rule:

$$P_M(Q) = \prod_{j=1}^{n_Q} P_M(x_j = q_j | x_{j+1} = q_{j+1}, \ldots, x_{n_Q} = q_{n_Q}), \tag{2}$$

and then apply the standard belief propagation algorithm [29] to each of the factors on the right-hand side of the Equation 2. For the case of binary data the time complexity of a single belief propagation in a Chow-Liu tree scales linearly in the number of edges in the tree [21], which is proportional to k. Thus, finding the answer to the query on n_Q attributes will scale as $O(n_Q k)$.

E. Mixtures of Independence (Bernoulli) Models

A probabilistic mixture model can be thought of as a generative model, i.e., a procedure for generating data under the assumption that it comes from N_c different groups or clusters. A data record is assumed to be produced by first selecting one of the N_c groups or clusters, where the probability of selecting cluster i equals $P_M(C_i)$, for $i = 1, \ldots, N_c$, and satisfies $\sum_{i=1}^{N_c} P_M(C_i) = 1$. Once cluster i is selected, a vector of attribute values is sampled from the probability distribution of the i-th cluster, $P_M(A_1, \ldots, A_k | C_i)$.

In the mixture of *independence* (Bernoulli) models there is an additional assumption in each cluster that the attributes are conditionally independent given the cluster so that

$$P_M(A_1, \ldots, A_k | C_i) = \prod_{j=1}^{k} P_M(A_j | C_i). \tag{3}$$

Thus, the mixture of independence models has the following functional form

$$P_M(A_1, \ldots, A_k) = \sum_{i=1}^{N_c} P_M(A_1, \ldots, A_k | C_i) P_M(C_i) = \sum_{i=1}^{N_c} \left(\prod_{j=1}^{k} P_M(A_j | C_i) \right) P_M(C_i). \tag{4}$$

The parameters of the distribution in Equation 4 that need to be estimated from the data are $\theta_{ij} = P_M(A_j = 1 | C_i)$ and $\lambda_i = P_M(C_i)$ for all $i = 1, \ldots, N_c$. Note that the mixture model avoids the problem of trying to directly specify a full dependence model which would require $O(2^k)$ parameters. Instead it seeks a more parsimonious representation such that within a specific cluster C_i the attributes are conditionally independent. Nonetheless, the attributes have marginal dependence imposed via the clusters C_i. A mixture model with N_c conditionally independent groups is fully specified by $O(kN_c)$ parameters. The number of clusters N_c is the tuning parameter of the model which we varied from 5 to 80 in our experiments.

Parameter estimation can be carried out using the Expectation Maximization (EM) algorithm which is an iterative process for likelihood maximization in the presence of hidden variables (which is the cluster variable in our case) [11]. The derivation of the EM update equations for the mixture of independence models follows the steps described in [13]:

$$\theta_{ij} = \frac{\sum_{l=1}^{n} P_M(C_i | \bar{A}_l) I(A_{lj} = 1)}{\sum_{l=1}^{n} P_M(C_i | \bar{A}_l)}, \ 1 \leq j \leq k;$$

$$\lambda_i = \frac{1}{n} \sum_{l=1}^{n} P_M(C_i|\bar{A}_l), \ 1 \leq i \leq N_c;$$

$$P_M(C_i|\bar{A}_l) = \frac{P_M(\bar{A}_l|C_i)\lambda_i}{\sum_{j=1}^{N_c} P_M(\bar{A}_l|C_j)\lambda_j}, \ 1 \leq l \leq n, \tag{5}$$

where n is the number of records in the data set and $A_{lj} \in \{0,1\}$ is the value of the j-th attribute in the l-th row, $\bar{A}_l$ is the l-th data record and the value of $P_M(\bar{A}_l|C_i)$ is obtained from Equation 3:

$$P_M(\bar{A}_l|C_i) = \prod_{j=1}^{k} \theta_{ij}^{A_{lj}}(1-\theta_{ij})^{1-A_{lj}}.$$

The process starts off from random initial values for the $P_M(C_i|\bar{A}_l)$ (say, randomly) and proceeds in an iterative fashion by first estimating λ_i and θ_{ij} (the M-step), then calculating $P_M(C_i|\bar{A}_l)$ (the E-step), and so on. A nice property of the EM procedure is that it is linear in the dimensions of the data set and the number of clusters, allowing for a low offline time cost for fitting the model. The time complexity per iteration is $O(N_c nk)$.

Once the mixture model is learned it is straightforward to calculate $P_M(Q)$ for any query Q. By using the attribute independence assumption we are able to sum out all but the query variables:

$$P_M(Q) = \sum_{\bar{x_Q}} P_M(x_Q, \bar{x_Q}) = \sum_{\bar{x_Q}} \sum_{i=1}^{N_c} \Big(P_M(x_Q|C_i) P_M(\bar{x_Q}|C_i)\lambda_i \Big) = \tag{6}$$

$$= \sum_{i=1}^{N_c} P_M(x_Q|C_i) \Big(\sum_{\bar{x_Q}} P_M(\bar{x_Q}|C_i) \Big)\lambda_i = \sum_{i=1}^{N_c} P_M(x_Q|C_i)\lambda_i = \sum_{i=1}^{N_c} \Big(\prod_{j=1}^{n_Q} \theta_{ij}^{q_j}(1-\theta_{ij})^{1-q_j} \Big)\lambda_i,$$

where $\bar{x_Q}$ is the complement to the set of query variables. Thus, online time is linear in the number of clusters and the size of the query: $t_P = O(n_Q N_c)$.

F. The Itemset Inclusion-Exclusion Model Based on ADTrees

In this section we describe how itemsets can be stored in an ADTree and subsequently used to answer queries via the inclusion-exclusion principle.

The ADTree is a sparse data structure that is useful for a variety of applications involving symbolic attributes, including fast counting, learning belief networks, association rule mining. Further details can be found in [22], [5], [25]. Ideas of storing the itemsets using tree structures have been used also in [34], [15].

We first store all itemsets in an array indexed by the size of the itemset. We then cycle thorough this array starting with itemsets of length 1 and add all the itemsets of the current size and their counts to an ADTree as follows. Adding an itemset of size 1 consists of adding a child to the root node of the ADTree whose only variable and the counts are the ones mentioned by the itemset. As was pointed out in the lemma of Section III an itemset I of size greater than 1 can only be frequent if the itemsets on all possible subsets of I are frequent as well. Thus the only action we need to take when adding itemset I is to link it to an appropriate parent-itemset in the tree and record its count.

Once the ADTree is learned one can use an efficient recursive procedure that implements the inclusion-exclusion principle to find the count of any conjunctive query:

$$c(x_1 = 0, \ldots, x_{n_Q} = q_{n_Q}) = c(x_2 = q_2, \ldots, x_{n_Q} = q_{n_Q}) - c(x_1 = 1, \ldots, x_{n_Q} = q_{n_Q}), \tag{7}$$

where $c(.)$ denotes the count of the expression in the brackets. In this manner the count of an arbitrary conjunctive query Q can be expressed as a signed sum of counts using only subsets of positively initialized attributes, which is exactly what the ADTree stores. Notice that there is a possibility that some of the terms in this signed sum may not have been retained since they were not T-frequent. The approach we follow in this paper is to discard such terms from the computation, i.e., assuming in effect that they are zero. However, other strategies are possible as well, such as estimating the values of missing terms. This estimation could be carried out in a variety of ways, for example, by using an independence model, a Chow-Liu tree model, a mixture model and so forth. Note, however, that such estimation will increase the time and memory complexity. Given that the "simple" inclusion-exclusion method described above (that ignores missing terms) turned out to be quite accurate across the range of simulated and real data sets that we investigated (as we will see later in the paper), we conjecture that the estimation of the additional terms need not necessarily increase the accuracy of the method very much. Thus, although more sophisticated inclusion-exclusion schemes could certainly be investigated, we consider them to be somewhat beyond the scope of this particular paper.

In the worst case (a query with all attributes initialized to 0's) the right-hand side of Equation 7 has $O(2^{n_Q})$ terms (after all terms are expressed in terms of T-frequent itemsets). Thus, the time complexity of answering a query Q is $O(2^{n_Q})$ in the worst-case.

G. The Maximum Entropy Method

In previous work we have investigated the maximum entropy method extensively in the context of query selectivity estimation ([17], [24], [25]). To save space we refer the reader to these earlier papers for complete details on the method and just briefly outline below the salient aspects of this model.

We assume that T-frequent itemsets and the associated frequency counts have been estimated offline (e.g., by the Apriori algorithm), and we view them as specifying constraints on the unknown distribution for the query variables $P_M(x_Q)$. The maximum entropy criterion is then used to select a unique probability distribution $P_M(x_Q)$ from the set $\mathcal{P}$ of all plausible distributions satisfying the constraints. If the constraints are consistent then the target distribution exists, is unique [6], and can be found in an iterative fashion using an algorithm known as iterative scaling [10]. The constraints generated by itemsets are consistent by definition since the empirical distribution of the data satisfies them.

Thus, the maximum entropy approach has the following general structure: (1) itemsets are learned offline, (2) a query is then posed in real-time, and (3) a joint probability distribution for the query variables is then estimated based on the itemsets alone. The learned distribution defines an *undirected* graphical model (a Markov random field or MRF) [25].

Inference of an undirected joint distribution on the query variables is performed by the iterative scaling algorithm. The algorithm has worst-case time complexity exponential in the number of query variables [29]. In [25] we considered several graph-based algorithms that reduce the time complexity of iterative scaling to being exponential in the *induced width* w^* of the graph of the MRF and we demonstrated that *the clique tree method* is the most efficient of these algorithms. In the results reported in this paper we use the clique tree method in our implementation of iterative scaling.

H. Summary of the Models

To complete this Section we summarize the worst case complexity of various models in the table I.

TABLE I

Worst-case Memory, Offline and Online Time Costs for Various Models. k is the number of attributes, n is the number of records, n_Q is the query size, N is the number of itemsets, n_j is the size of the j-th itemset, N_c is the number of clusters in the mixture of independence models, p.i. stands for "per iteration".

Group	Model	Memory	Online Time	Offline Time
	Linear Scan	$O(\sum_{i=1}^n N_{1's}(i))$	$O(n_Q \sum_{i=1}^n N_{1's}(i))$	$O(1)$
	Sample, $t\%$	$O(t \sum_{i=1}^n N_{1's}(i))$	$O(tn_Q \sum_{i=1}^n N_{1's}(i))$	$O(1)$
1	Independence	$O(k)$	$O(n_Q)$	$O(\sum_{i=1}^n N_{1's}(i))$
	Bernoulli Mixture	$O(kN_c)$	$O(n_Q N_c)$	$O(knN_c)$, p.i.
	Chow Liu	$O(k^2)$	$O(kn_Q)$	$O(k^2 n)$
2	ADTrees	$O(\sum_{j=1}^N n_j + N)$	$O(2^{n_Q})$	$O(N \sum_{i=1}^n N_{1's}(i))$
	Maximum Entropy	$O(\sum_{j=1}^N n_j + N)$	$O(N^2 2^{n_Q})$, p.i.	$O(N \sum_{i=1}^n N_{1's}(i))$

As shown in the table, there are various tradeoffs in the worst case between the memory required for a model, the online time cost for querying, and the offline construction time. For example, a linear scan of the full data is the worst in terms of memory requirements, it takes significant online time, but it has the least offline time cost (i.e., zero, since no model is created). Note that all methods fall into one of two groups: (1) those estimated once offline on all of the attributes in the data set (group 1 in the table), and (2) those based on the itemsets that estimate online a distribution on the specific variables in a given query (group 2 in the table). In the next section we empirically investigate the average time and memory performance of each of these approaches, and in addition evaluate the empirical errors (Equation 1) for each model.

V. Empirical Results

A. Description of the Data Sets

We conducted experiments on two real-world transaction data sets, as well as on sets of simulated data. The real-world data sets included the Microsoft Anonymous Web data set (publicly available at the UCI KDD archive, kdd.ics.uci.edu) with 32,711 records (Web site visitors) and 294 fields (Web pages), and a large proprietary data set of consumer retail transactions with 54,887 records (transactions) and 52 fields (categories of items that can be purchased). The simulated data sets were designed to be much larger than these real-world data sets.

General characteristics of these data sets are provided in the Table II. The retail data set is much more dense than the Microsoft Web data as indicated by the *density* index which is defined as the probability that a 1 appears in any randomly selected cell in the data matrix. The density index for the retail data is almost 8 times higher than that for the Microsoft data. As data density grows, the larger itemsets become more frequent and the memory footprints of the itemset-based maximum entropy and ADTree models will also increase. Furthermore, the graphs underlying the maximum entropy probability model also become dense, leading to potentially significant increases in the online estimation time of the maximum entropy approach. We will see empirical examples of these effects later in this section.

To analyze the structure of the frequent itemsets for each data set we considered different values of the threshold T and counted the number of T-frequent itemsets in the data. Table III gives the distribution of the number of T-frequent itemsets of size s for different values of s and T. The last

TABLE II

GENERAL CHARACTERISTICS OF THE DATA SETS: k IS THE NUMBER OF ATTRIBUTES, n IS THE NUMBER OF RECORDS, $N_{1's}$ IS THE NUMBER OF 1'S IN THE DATA, $E(N_{1's}) = N_{1's}/n$, $E(N_{1's})/k$ IS THE DENSITY INDEX, $Std(N_{1's})$ IS THE STANDARD DEVIATION OF THE NUMBER OF 1'S IN THE RECORD, AND $Max(N_{1's})$ IS THE MAXIMUM NUMBER OF 1'S IN ANY RECORD.

	k	n	$N_{1's}$	$E(N_{1's}/k)$	$Std(N_{1's})$	$Max(N_{1's})$
MS Web Data Set	294	32711	98654	0.0102	2.5	35
Retail	52	54887	224580	0.0786	3.98	44

row provides a heuristic estimate of the induced width w^* of the graphs of the model, using the maximum cardinality ordering d on all attributes. The largest clique in the triangulated graph of the retail data set mentions roughly the half of all variables, while the same parameter for the Web data is only 10% of the number of all variables, despite the order of magnitude difference between the corresponding thresholds T. Thus, even for queries on the retail data that mention only 6 to 8 variables, we can expect to get much more dense graphs using the maximum entropy method and correspondingly higher online query times.

TABLE III

DISTRIBUTION OF THE ITEMSETS FOR THE WEB DATA AND THE RETAIL DATA. THE LAST ROW PROVIDES A HEURISTIC ESTIMATE OF THE INDUCED WIDTH w^* IN THE GRAPH FOR ALL ATTRIBUTES.

size	MS Web		Retail	
s	T=20	T=30	T=300	T=400
2	183	107	1486	1377
3	1453	1041	175529	131329
4	3102	1891	3904616	1995004
5	3178	1576	17650712	5737579
6	1542	595	24580567	4937654
7	381	134	14180476	1401976
8	56	20	2717947	85906
9	6	1	133387	596
10	0	0	742	1
$w^*(d)$	30	24	25	23

B. Query Generation

We empirically evaluated the following models
1) the training data;
2) $q\%$ samples of the training data for values of $q = 1, 5, 10, 15, 20$;
3) the independence model;
4) the Chow-Liu tree model;
5) the mixture of independence models parameterized by the number of clusters N_C, taking values $5, 10, 25, 50, 80$;
6) the itemset inclusion-exclusion model parameterized by the value of the threshold T in the definition of T-frequent itemsets. This parameter was chosen differently for different data sets because of the differences in their density. For the Microsoft Web data, we set T to $10, 30, 50, 70, 90$ and for the retail data, we set to T to $200, 300, 400, 500, 600$;

7) the maximum entropy method using clique-tree iterative scaling, parameterized by a threshold T in the definition of T-frequent itemsets. The threshold T took the same values as above.

Of special note here is the fact that the number of clusters N_C (in the definition of the mixture models) and the threshold T (in the definition of itemset-based methods) allow a direct tradeoff of accuracy for time and memory. In particular, the higher the number of clusters N_C, the more parameters are in the model (linearly as a function of N_C) and the more accurately it can answer queries on the training data. But as N_C increases it also requires more memory. Similarly, as the threshold T is reduced, more itemsets become T-frequent and are used in the model. Thus, for smaller values of T, the itemset-based models use more memory and presumably offer more accurate estimates than for larger values of T. In the following sections we verify these conjectures empirically.

All experiments were performed on a Pentium III, 450 MHz machine with 128 Mb of memory.

We generated 1000 queries from a fixed user query distribution $\pi(Q)$, and evaluated different models with respect to the average memory taken by the model, the average online time taken to answer a query, and the average empirical error as defined in Equation 1.

The distribution $\pi(Q)$ was modelled as follows. We first fixed its size n_Q to 4 or 8. This choice reflects our assumption that users querying the data are more likely to ask short queries so that $\pi(Q)$ is peaked at small n_Q's. Once the value for n_Q was chosen, n_Q attributes were randomly selected according to the probability of the attribute taking a value of "1" and a value for each selected attribute was generated randomly according to its univariate probability distribution. This choice of a user query distribution $\pi(Q)$ is motivated by the fact that zero values for the attributes are more likely in sparse data sets than positive ones. Using purely random queries (randomly chosen attributes with randomly chosen values) would result in a preponderance of queries whose count is zero in the data (since any query consisting of more than one positively instantiated attribute will often not have occurred in the data).

C. Results on the Microsoft Web Data

The left plots in Figure 1 show the the average relative error versus the memory requirements for each model, using the Microsoft Web data. The top plots correspond to queries of size 4, the bottom to queries of size 8. Both axes on each plot are on a logarithmic (base 10) scale. The dotted line corresponds to a linear scan of the training data—it is error free, so that the logarithm of error equals minus infinity, and thus, we only show how much memory the training data takes. Note that all models to the right of this dotted line are effectively impractical since that they are taking more memory than the full data set. Different points on each curve were obtained by varying the value of the tradeoff parameter for the respective model plotted by that curve. The values of the tradeoff parameters for each model, and their influence on the model's complexity and accuracy, were summarized in Section V-B. For instance, lower values of the parameter T (for itemset-based models) result in a larger number of itemsets, larger memory requirements, and (typically) lower error rates.

The independence model requires the least memory but it is also the most inaccurate model. Since the Chow-Liu tree model incorporates specific pairwise dependencies between attributes, it is more complex (taking more memory than the independence model) and exhibits better accuracy than the independence model. However, the accuracy of the Chow-Liu tree model is not as good as the mixture, itemset inclusion-exclusion, or maximum entropy models. The query answers from random samples of various sizes are comparable in accuracy to the independence and Chow-Liu

models. However, they are less accurate then the other models, although they take similar memory to the mixtures (for example). As we noted above the larger the number of clusters N_C, the more parameters are in the mixture model (linearly as a function of N_C) and the more accurately it can answer queries on the training data. This is verified by the Figure 1. The itemset-based inclusion-exclusion ADtree and maximum entropy models are the most memory-intensive but they are also the most accurate. The model based on maximum entropy is the most accurate overall for most values of the threshold T.

We also measured the average online time taken to generate the estimated query count. Right plots on Figure 1 illustrate how the error depends on the online time for all of the models answering queries of length 4 (top) and 8 (bottom) on the Web data.

The independence model is the fastest but it is also the least accurate of all models. The Chow-Liu model is slightly more accurate than the independence model but takes roughly an order of magnitude more time. The most accurate of the itemset inclusion-exclusion models are extremely fast (about as fast as the independence model). The itemset inclusion-exclusion models are not as accurate as the models based on maximum entropy on short queries of length 4, but approach the accuracy of the maximum entropy models on longer queries of length 8. The best mixture models ($N_C = 80$) are not as accurate as the best itemset models (based on either inclusion-exclusion or maximum entropy), but are closer in speed to the fast itemset inclusion-exclusion models than to the relatively slow model based on maximum entropy.

The itemset model based on maximum entropy is more accurate than all other models but takes orders of magnitude more time to produce estimates. In absolute terms, queries of size 4 are often being answered in well under a CPU second, but relative to the other models this is quite slow. The time requirements to answer queries of size 8 are greater than the time to perform a linear scan of the training data, regardless of the value chosen for T.

As the query size changes, from queries of length 4 (top right plot in Figure 1) to queries of length 8 (bottom right plot), all of the methods except for maximum entropy appear relatively insensitive to query length. Maximum entropy is highly sensitive, however, since it builds a graph-based model for the query variables in real-time. As the query size increases (from top to bottom in the plots) the average online time for the model using maximum entropy increases dramatically, as the average induced width in the graphs for the query attributes increases accordingly.

D. Results on the Retail Data

Since this data set is more dense, we increased the threshold T for the itemset models (inclusion-exclusion and maximum entropy) as described in Section V-B. All of the remaining models (the independence, the Chow-Liu, and the mixture models) were constructed in the same fashion as for the Web data.

The memory sizes of the maximum entropy and itemset inclusion-exclusion methods increased by about an order of magnitude on this data compared to the Web data (compare left plots on Figures 1 and 2). The other models (independence, Chow-Liu tree, and mixtures) actually tended to *decrease* in memory size compared to the Web data since they are only a function of the number of variables in the data and not a function of the density.

The accuracy of the various models relative to one another on the retail data was qualitatively similar to that on the Web data with one significant exception, namely that the mixture model now outperformed maximum entropy and became the most accurate model overall among models considered in the experiment (see Figure 2). In theory we could likely find an itemset model using maximum entropy that would outperform the best mixture model, by lowering the threshold T

(i.e., including more itemsets). However, this would incur a very significant computational penalty in terms of memory and online computation time. For queries of size 8 the average CPU time per query is already over 1 second, and for $T = 200$ the memory is over 1MByte. Similarly, for the itemset inclusion-exclusion models, the accuracy could be improved by allowing the model to take up more memory. However, for this data set this would again be impractical. Thus, on the more dense data neither the itemset inclusion-exclusion nor the maximum entropy model can match the overall performance of the mixture model given reasonable memory constraints.

E. General Observations

From the results on both data sets a number of general observations can be made:

- The independence models are the fastest, have the smallest memory requirements, but are the least accurate of all models.
- The Chow-Liu tree model provides a modest improvement in accuracy over the independence model but is slower and requires more memory.
- Random samples are substantially less accurate than the mixture, inclusion-exclusion, and maximum entropy methods, and in addition they can take significant time and memory resources.
- The itemset inclusion-exclusion models are much faster than all of other models (except for the independence model). The largest itemset inclusion-exclusion model can be comparable in accuracy with the best of the other methods. However, on dense data sets the itemset inclusion-exclusion models require large memory investments in order to be one of the best models.
- The itemset models based on maximum entropy are among the most accurate models, but suffer from an exponential increase in online time as the query length grows, and they suffer the same memory problems as the itemset inclusion-exclusion models.
- The mixture models tend to provide comparable accuracies to those of the itemset inclusion-exclusion and the maximum entropy methods (beating them on the dense data and losing to them on the sparse data). However, they are much faster than the models based on maximum entropy and have a much smaller memory footprint than both types of itemset-based models. Thus, mixture models offer a quite useful operating point in the memory-speed tradeoff space.

F. Scalability to Large Data Sets

The Microsoft web and retail data sets used in our experiments above are relatively small compared to many real-world transaction databases. The experiments were useful and informative in that they provided a relative evaluation of different approximation schemes. However, from a practical viewpoint it is of interest to know how the models scale up to data sets that are much larger. To investigate this we ran experiments in the following manner. We first learned a mixture model (Equation 4) with 20 clusters on the Web data and then simulated data sets with 30K, 300K and 3000K records from the learned mixture model. The sample of size 30K is roughly equal to the size of original data, while the other two samples are correspondingly 10 and 100 times larger. For the maximum entropy method we set the value of the threshold T to 30, 300 and 3000, for the 30K, 300K, and 3000K sized data sets respectively, and computed T-frequent itemsets for each of the samples.

Our goal here was to evaluate how speed and memory characteristics of the different methods changed as the data size was increased. By simulating from a non-trivial model (i.e., not just

an independence model) built from real data, the resulting simulated data can be expected to be similar in many respects to a real data set of the same size, e.g., relatively large itemsets can be generated and/or records with large number of 1's.

We investigated how the maximum entropy method enhanced with bucket elimination computation (see [25] for details), the itemset inclusion-exclusion, and the mixture models compared to a linear scan of the data in terms of both memory and online time when the size of the training data increased. We ignored accuracy in our experiments since on simulated data it is not particularly meaningful. The memory for the maximum entropy and itemset inclusion-exclusion models built for each of three data sets was approximately 0.1Mbit. This compares with 0.43Mbit memory for storing the whole dataset with 30K records, 4.6Mbits for the data with 300K records, and 46Mbits for 3000K records. Table IV describes the online timing results. As expected, the time of a linear scan increased linearly with the size of the data set. Given our choice of the threshold values the average online times of the maximum entropy and itemset inclusion-exclusion methods, however, remained roughly constant as the size of the data set increases. For the largest data set with 3 million records, the maximum entropy method was two orders of magnitude faster than a linear scan for queries of length 4 and similar in speed to a linear scan for queries of length 8. For a given fixed-size data set, as query length increases, the maximum entropy method will eventually become slower than a linear scan of data (again a function of the exponential growth in time complexity as a function of query length for the maximum entropy approach).

The mixture model times was also included for comparison in Table IV. The times were independent of the data size (since the model is independent of the data size) and varied linearly with query length. In absolute terms the mixture model was significantly faster than either the maximum entropy approach or the linear scan. For large real-world data sets the mixture model or the itemset inclusion-exclusion model (if memory is plentiful) may well be the techniques of choice for fast and accurate approximate querying.

TABLE IV

AVERAGE ONLINE TIMES (IN CPU SECONDS) FOR VARIOUS SIMULATED DATA SAMPLES AND QUERY SIZES. THE FOLLOWING ABBREVIATIONS ARE USED BELOW: ME—THE ITEMSET MODEL USING MAXIMUM ENTROPY, MM–A MIXTURE MODEL WITH $N_C = 100$ COMPONENTS, IIE IS AN ITEMSET INCLUSION-EXCLUSION MODEL, AND SS–A LINEAR SCAN THROUGH THE DATA.

Query Size	Algorithm	Data Size		
		30K	300K	3000K
	ME	0.054	0.051	0.049
4	IIE	0.00014	0.00025	0.00061
	MM	0.002	0.002	0.002
	SS	0.022	0.236	2.422
	ME	2.271	1.534	1.631
8	IIE	0.00071	0.00068	0.00072
	MM	0.006	0.006	0.006
	SS	0.029	0.281	2.792

VI. CONCLUSIONS

We have investigated the application of general probabilistic models to the problem of query approximation for binary transaction data sets. We introduced two new methods for building prob-

ability models from frequent itemsets: a maximum entropy approach and an inclusion-exclusion approach. Several other probabilistic modelling techniques were also investigated and compared to the linear scan of the original data and random samples of original data: the independence model (as a baseline), Chow-Liu tree models that incorporate specific pairwise attribute dependencies, and mixtures of Bernoulli (independence) models.

In a variety of experiments on real-world and simulated transaction data we found that virtually all of these models provided gains in accuracy over the simple independence model. However, each came with an associated cost. The itemset inclusion-exclusion model provides improved accuracy and very fast online computation times, but at a cost of having quite large memory footprints to store a large number of itemsets in an ADTree. The itemset model using maximum entropy has the same memory requirements and is even more accurate than the itemset inclusion-exclusion model, but on each iteration of the model-fitting procedure it scales exponentially in time as a function of query length. Mixture models appear to offer a useful practical trade-off by avoiding both the excessive memory and time requirements of the maximum entropy and itemset inclusion-exclusion methods, while still providing substantial improvements in accuracy over the baseline independence model. Generating approximate predictions by scanning random samples of the original data tends to be too time or memory consuming, or inaccurate, and appears to be less effective than the model-based approaches.

Which of these methods is best-suited for a particular application is a function of the time and space resources available for the application as well as the nature of the queries being issued and the nature of the underlying data. All of the probabilistic models we considered in this paper (except for independence and Chow-Liu trees) have parameters that can be tuned to satisfy specific end-user requirements regarding the trading of time and memory for accuracy.

In this context, we have recently investigated the use of model-combining algorithms that can use a weighted combination of multiple models to answer a query [27]. The relative weights for different models are chosen in an adaptive and automated manner to optimize accuracy relative to a given query distribution and a given data set. The algorithm for combining models uses a straightforward and scalable least-squares optimization procedure. The use of model-combining in this manner sidesteps the issue of having to select a single "best" model by adaptively using combinations of models. In fact, on real-world and simulated data sets, the predictions from the weighted combination of models reduce the prediction error of any single model by factors of up to 50% [27].

We emphasize that, unlike most previously reported work on approximate querying, our models are able to handle sparse high dimensional data sets with numbers of attributes on the order of hundreds or thousands.

A variety of extensions of each of the probabilistic models discussed in this paper are possible. For example, it is relatively straightforward to extend each of the probability models to answer queries expressed as arbitrary Boolean functions rather than as simple conjunctions. The methods can also be extended to arbitrary categorical data (or discretizations of real-valued data) rather than being restricted to binary data alone. Anytime algorithms for approximate query-answering are also of significant practical interest. In [28] we present a more detailed description of the framework and its extensions.

References

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The AQUA approximate query answering system. In *Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 574–576. New York, NY: ACM Press, 1999.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207–216. New York, NY: ACM Press, 1993.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. Menlo Park, CA: AAAI Press, 1996.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499. San Francisco, CA: Morgan Kaufmann Publishers, 1994.

[5] B. S. Anderson and A. W. Moore. ADtrees for fast counting and for fast learning of association rules. In *Proceedings Fourth International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM Press, 1998.

[6] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72, 1996.

[7] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *The VLDB Journal*, 3:111–122, 2000.

[8] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3):462—467, 1968.

[9] T.H. Cormen, C.E. Leiserson, and R.R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[10] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B-39:1–38, 1977.

[12] A. Deshpande, M. Garofalakis, and R. Rastogi. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'01)*, pages 199–210. New York, NY: ACM Press, 2001.

[13] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[14] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'01)*, pages 461–473. New York, NY: ACM Press, 2001.

[15] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'00)*, pages 1–12. New York, NY: ACM Press, 2000.

[16] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 19(2):1–11, 1990.

[17] H. Mannila, D. Pavlov, and P. Smyth. Predictions with local patterns using cross-entropy. In *Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'99)*, pages 357—361. New York, NY: ACM Press, 1999.

[18] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings of Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'96)*, pages 189–194. Menlo Park, CA: AAAI Press, 1996.

[19] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181–192. Menlo Park, CA: AAAI Press, 1994.

[20] Y. Matias, J. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 448–459. New York, NY: ACM Press, 1998.

[21] M. Meila-Predoviciu. *Learning with mixtures of trees*. PhD thesis, Massachusetts Institute of Technology., 1999.

[22] A. W. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

[23] M. Muralikrishna and D. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'88)*, pages 28–36. New York, NY: ACM Press, 1988.

[24] D. Pavlov, H. Mannila, and P. Smyth. Probabilistic models for query approximation with large sparse binary data sets. In *Proceedings of the Uncertainty in AI conference (UAI'00)*, pages 465–472. San Francisco, CA: Morgan Kaufmann Publishers, 2000.

[25] D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. Technical Report UCI-ICS-TR-01-09, Information and Computer Science, University of California, Irvine, 2001.

[26] D. Pavlov and P. Smyth. Probabilistic query models for transaction data. In *Proceedings of Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'01)*, pages 164–173. ACM Press, 2001.

[27] D. Pavlov and P. Smyth. Adaptive approximate querying of large sparse binary data sets via probabilistic model averaging. Technical Report 2002-050, NEC Research Institute, May 2002.

[28] D. Y. Pavlov. *Probabilistic Query Models for Transaction Data*. PhD thesis, Department of Information and Computer Science, University of California Irvine., 2002.

[29] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[30] V. Poosala and Y. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 486–495. San Francisco, CA: Morgan Kaufmann Publishers, 1997.

[31] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[32] J. Shanmugasundaram, U. Fayyad, and P. Bradley. Compressed data cubes for OLAP aggregate query approximation on continuous dimensions. In *Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'99)*, pages 223–232. New York, NY: ACM Press, 1999.

[33] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'99)*, pages 193–204. New York, NY: ACM Press, 1999.

[34] D-Y. Yang, A. Johar, A. Grama, and W. Szpankowski. Summary structures for frequency queries on large transaction sets. In *Proceedings of Data Compression Conference*, pages 420–429. Snowbird, UT: IEEE Computer Society Press, 2000.

Fig. 1. Average relative error for 1000 queries of length 4 (top) and 8 (bottom) drawn from $\pi(Q)$ as a function of the average model complexity (left) and average online time (right) for the Web data.
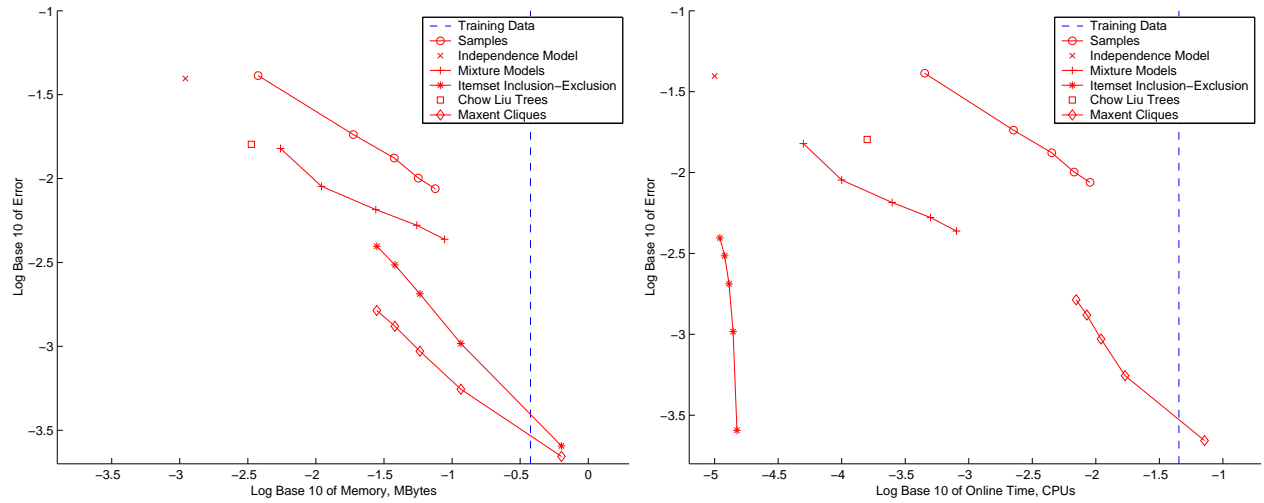
(a) $n_Q = 4$
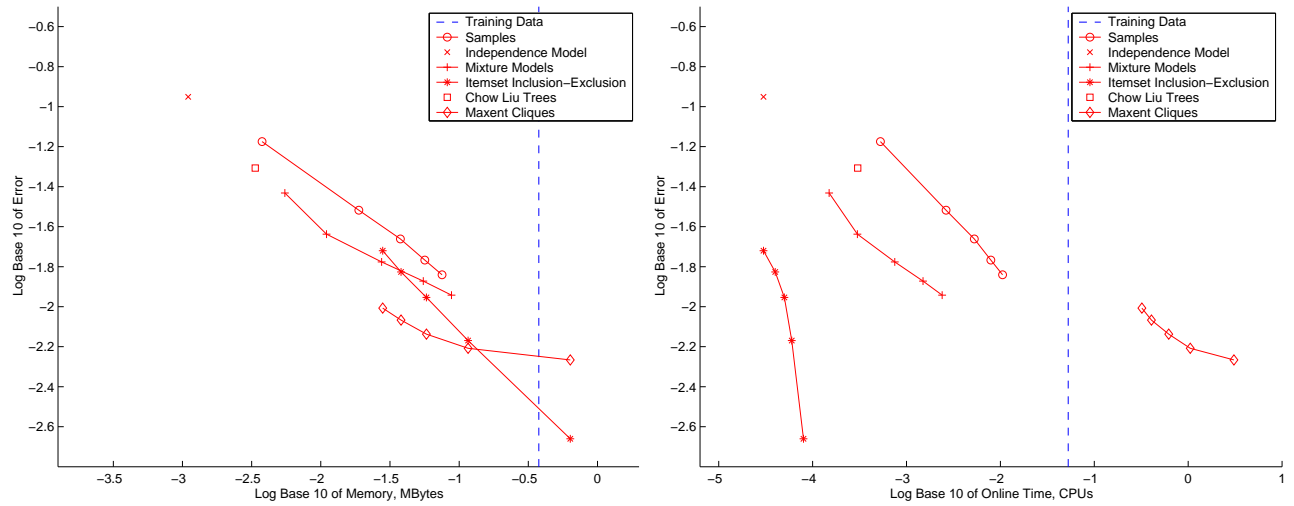
(b) $n_Q = 8$

Fig. 2.   Average relative error for 1000 queries of length 4 (top) and 8 (bottom) drawn from $\pi(Q)$ as a function of the average model complexity (left) and average online time (right) for the retail data.

(a) $n_Q = 4$

(b) $n_Q = 8$