Tractable Learning of Large Bayes Net Structures from Sparse Data

Anna Goldenberg ANYA@CMU.EDU
Center for Automated Learning and Discovery, CMU 5000 Forbes Ave, Pittsburgh, PA 15213 USA

Andrew Moore AWM@CS.CMU.EDU
Robotics Institute, CMU 5000 Forbes Ave, Pittsburgh, PA 15213 USA

Abstract

This paper addresses three questions. Is it useful to attempt to learn a Bayesian network structure with hundreds of thousands of nodes? How should such structure search proceed practically? The third question arises out of our approach to the second: how can Frequent Sets (Agrawal et al., 1993), which are extremely popular in the area of descriptive data mining, be turned into a probabilistic model?

Large sparse datasets with hundreds of thousands of records and attributes appear in social networks, warehousing, supermarket transactions and web logs. The complexity of structural search made learning of factored probabilistic models on such datasets unfeasible. We propose to use Frequent Sets to significantly speed up the structural search. Unlike previous approaches, we not only cache n-way sufficient statistics, but also exploit their local structure. We also present an empirical evaluation of our algorithm applied to several massive datasets.

Keywords: Bayesian networks/graphical models, statistical learning, Bayes Net structure learning

1. Introduction

Bayesian Networks have been successfully applied in areas such as pharmaceutical research, decision making by doctors, air control and marketing. Structural learning of Bayesian Networks is usually desirable but costly.

This paper provides an algorithm for tractable structural learning in Bayes Nets by exploring structures on the local level. We exploit the computational efficiency of Frequent Sets for gathering statistics that are most likely to be useful for structure search given the assumption of sparse data. We then give an efficient search algorithm to exploit these statistics for creating the global Bayes Net.

1.1. Why learn large Bayes Nets?

Usage of Bayesian Networks to represent expression of genes based on the activity of their regulators (in practice approximated by protein activity levels) is well motivated by Friedman (2004). He suggests that the structure of the network is important in itself, since it may provide information about gene regulators.

Another field that has received increasing attention in the last few years is recommender systems. Online systems such as Amazon provide suggestions of what might appeal to the user based on user's other preferences. The use of Bayesian networks in this domain has become established, e.g. (Breese et al., 1998). Often the goal of recommender systems is to predict which are the most likely items that the user would buy next. An example of answering analogous query using Bayes Nets built by our algorithm is presented in Section 6.2.

The idea of representing social networks as people connected by directed arrows has been explored in social science domain for almost 70 years (Moreno & Jennings, 1938). Initially analyzed networks were on the order of 10s of nodes. However, improvements in data collection and especially the birth of online communities made it necessary to look at much larger networks. For example, livejournal (an online blog community) contains over 50,000 users (Shklovsky, personal communication). Usage of graphical models in this domain has become increasingly popular, due to their robustness to noise.

Studies in the gene expression data and social networks in particular suggest that correlations of entities on the local level are very important and in fact they are what makes up the global network (Friedman, 2004; Breiger, 2003). So, along with being computationally practical, Bayesian Networks created by our algorithm have a very natural motivation stemming from those important domains.

We provide results on sparse massive datasets showing practical training times, and in many cases superior ability to model the joint distribution in comparison with di-

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

rect extensions of traditional structure search algorithms on large data. We also qualitatively and empirically show that sparse data (particularly data with social net characteristics) are modelled better by going beyond information derived from pairwise co-occurrences.

2. Frequent Sets

Assume our training data is a collection of N records with M binary categorical attributes per record. Write x_{ij} as the value of the jth attribute of the ith record where $1 \leq i \leq N$ and $1 \leq j \leq M$. We assume sparse data in which the vast majority of values in any dataset row or column are zero. Note we assume no missing values: all x_{ij} are observed.

Let the M attributes be represented by integers $\{1, 2, \ldots M\}$. Let the *co-occurrence frequency* of a set of attributes $S \subseteq \{1, 2, \ldots M\}$ be the number of records in which all the attributes in S are simultaneously set to 1.

$$freq(S) = |\{i : \forall j \in S, x_{ij} = 1\}| \tag{1}$$

Given $s \geq 1$ we say S is a *Frequent Set of* m *attributes* if S contains exactly m attributes and $freq(S) \geq s$. Threshold s is called *support* in the data mining literature. Given sparse data and a support s greater than about 3, it is surprisingly easy to compute all Frequent Sets (Agrawal & Srikant, 1994). There is an abundance of literature on Frequent Sets as their collection is an essential part of the association rules algorithms (Agrawal et al., 1993; Agrawal & Srikant, 1994; Han & Kamber, 2000) widely used in commercial data mining.

There are multiple references to Frequent Sets in the area of modelling sparse datasets as well (Mannila & Toivonen, 1996; Chickering & Heckerman, 1999; Pavlov et al., 2003; Hollmen et al., 2003). This is not surprising, since sparseness implies very few co-occurrences between items. In fact, most items do not co-occur with each other, hence we expect the majority of the counts in the pairwise marginals to be 0. Therefore, it is natural to assume that the Frequent Sets contain most of the essential information about the whole dataset.

(Chickering & Heckerman, 1999) propose and show how to use an efficient sparse representation for several classes of machine learning algorithms including structure initialization for Bayes Nets. We will therefore not focus on representational aspects of Frequent Sets.

This paper exploits previous research on the utilization of Frequent Sets for modelling of sparse datasets but takes a new perspective. Assuming that Frequent Sets comprise essential information about our data we propose to exploit them to find Bayes Net structures on the local level. To our knowledge, structures contained *within* Frequent Sets have not been previously used in order to improve the global model of data.

3. Algorithm Description

The simplest idea for exploiting Frequent Set information is to use frequent pairs. The only edges which we would consider including in the Bayes Net are those for which the source and destination attributes co-occur more than some support s. There are thus far fewer edges to consider than the full $M(M-1)$ possibilities (M is the number of attributes).

There are three problems with this idea.

- **Problem 1.** This method will not find edges that have negative correlations. For example, if attribute A is never positive when attribute B is positive then (A,B) will not be a frequent pair and so will not be considered.
  Solution. Problem 1 is mitigated in two ways. First, under the assumption of sparse data there must necessarily be less evidence for a strong negative correlation as is shown in the Appendix. In fact, the structure scoring metric (e.g. BDeu) will be much higher for positively correlated entities. Secondly, attributes with high marginal positive values (where a negative correlation might be significant), will be accounted for at a later stage described in Section 4.
- **Problem 2.** Some items that do co-occur might be independent. This is especially likely with promiscuous attributes that occur frequently by themselves and thus could co-occur just by chance.
  Solution. The solution to problem 2 is to screen all frequent pairs before allowing links between them into the pool of edges considered for the network. Only significantly correlated pairs become candidate edges. This greatly reduces the number of candidate edges.
- **Problem 3.** Restricting the search to frequent pairs can miss significant higher-order interactions. The appendix gives one example, but it is easy to imagine many cases in which co-occurence of A and B is predictive of the occurence of X and yet one or both of the $A \to X$ and $B \to X$ dependencies are not statistically significant.
  Solution. This is solved by using higher-order Frequent Sets, as described in the following paragraphs.

Screening the Frequent Sets. We call the set of edges that will eventually be considered for addition into the Bayesian Network the *Edgedump*. Suppose we have a collection of Frequent Sets $\{X : |X| = m, m \geq 2\}$. First, we screen the pairs to find positive pairwise correlations. We add an edge between two variables to the *Edgedump* if and only if a significant correlation was found between the 2 variables in the pair. We then in turn screen for dependencies in Frequent Sets of size 3, 4, etc.

When does a Frequent Set X of size $m > 2$ provide new information valuable for building a Bayes Net? It is possible that the dependencies of X are already well-explained by interactions of order less than m. For example, suppose attributes A, B and C co-occur frequently, but their co-occurence is well explained by the local Bayesian Network DAG structure of $A \leftarrow B \rightarrow C$. In that case the two-way interactions will already explain all dependencies of X. In this case, X should not be added to the edgedump. In fact, only DAGs that contain a node with $m-1$ parents could be missed by considering only lower order interactions.

We implement a *Screening* test by searching over all possible DAG structures for X and finding whether the best BDeu-scoring structure (see Section 6) has an $m-1$-parent node (we call it an *m-way* interaction). We thus allow X to pass the screening test *if and only if* X is best explained by a local DAG structure containing an m-way interaction. If X passes the Screening test, all edges of the highest scoring DAG are added to the Edgedump.

Once the Edgedump is created, we prioritize the edges according to their strength, measured by the number of the m-way interactions in which they participate. We then create an empty (edgeless) global Bayesian network and iterate through the Edgedump contents, allowing each edge in turn to be added if and only if it improves BDeu and avoids cycles.

Table 1 contains the full description of the algorithm.

Table 1. Screen-based Bayes Net Structure search (SBNS) algorithm

algorithm	SBNS
input	**K** - max Frequent Set size
	s - support
output	BN - Bayes Net

Also: Ed - Edgedump - a collection of directed edges represented as (source,dest,count)
DS - DAG storage

1. **for** k = 2 .. K
2. obtain counts for all Frequent Sets of size k
3. **foreach** Frequent Set
4. find best scoring DAG
5. **if** DAG contains a node that has $k-1$ parents
6. store DAG in DS
7. **end foreach**
8. **end for**
9. **foreach** DAG in DS
10. store all edges $\{source, dest, count{+}{+}\}$in Ed
11. order Ed in decreasing order of edge counts
12. **foreach** edge $e \in Ed$
13. **if** e doesn't form a cycle in BN
14. **and** e improves BDeu
15. add e to BN
16. **end foreach**
17. return BN

4. Addition of High Mutual Information Links

In the previous section we pointed out that Frequent Sets bias the algorithm in favor of interactions that cause co-occurrence (and thus positive correlation). Appendix 1 shows why, in the case of sparse data, positive correlations must be stronger than negative correlations, so in general we are not omitting the strongest correlations. There is, however, still a danger that if a few attributes are promiscuous (relatively high univariate marginal probability, though still very sparse), they could cause significant negative correlations that we could miss. Fortunately, such negative pairwise correlations can be detected cheaply using a technique from (Meila, 1999).

Let I_{AB} be the mutual information between two attributes. Meila showed that the mutual information can be calculated in a very efficient manner, particularly when dealing with discrete binary data. In fact, if the two variables have not co-occurred in the dataset, the formula simplifies even further: I_{AB} is directly proportional to the magnitude of A (N_A) and B (N_B) as shown in Equation (2). The full derivation is available in (Meila, 1999).

$$\begin{aligned} I_{AB} &= H_A + H_B - H_{AB} \\ &= \frac{1}{N}\Big[-(N-N_A)log(N-N_B) - (N-N_B) \\ &\quad \times log(N-N_B) + (N-N_A-N_B) \\ &\quad \times log(N-N_A-N_B) + NlogN\Big] \end{aligned} \tag{2}$$

Hence, to add high mutual information edges, we have to check entities that occur with high frequency. We reduce the total number of entities significantly by only considering ones that occurred more than s times in the dataset. This step is statistically justified because fewer occurrences mean lower possible mutual information. Table 2 describes the algorithm that augments a given Bayes Net with high mutual information (MI) edges.

Table 2. Algorithm that augments Bayes Net BN with high MI edges

```
algorithm  AugmentWithMutualEdges
input      BN - a Bayes Net
           L - list of attributes with frequencies

Sort L in decreasing order of frequencies
for u = 1 ... |L| - 1
    v = u + 1; added_flag = TRUE
    while v < |L| & added_flag
        if net with e_uv score > old net score
            add e_uv to BN
        else try to add e_vu to BN
        v = v + 1
        if (edge not added) added_flag = FALSE
    end while
end for
return BN
```

Note that we do not have to search the space of all edges to find edges with the highest mutual information. First of all, we sort entities in descending order of frequency. For each entity $A_i, i = 1 \dots N_{>s}$, where $N_{>s}$ is the number of entities with support $> s$, we only consider $\{B_{ij}, j = i{+}1 \dots N_{>s}\}$, i.e. those entities that have occurred less frequently than A_i. If an edge $e_{A_i B_{ij}}$ has been rejected, then we move along the A list. This step is justified, because entities are sorted in descending order of frequencies, hence the mutual information between A_i and B_{ij+1} is lower than between A_i and B_{ij}. Thus, the edge $e_{A_i B_{ij+1}}$ is even less likely to be added than $e_{A_i B_{ij}}$. Empirical evidence shows that on average only 10% of N pairs are considered.

5. Additional possible postprocessing

5.1. Second Degree Separation Links

It is cheap to do an extra pass of edge-additions in which we iterate over all nodes in the network produced by the previous steps and attempt adding edges directly from the current node to its grandchildren.

5.2. Hillclimbing

One of the standard techniques to improve the score is hillclimbing as described in (Cooper & Herskovits, 1991). This technique improves the score by adding/removing/reversing arcs in a Bayes Net. The set of operations and edge selection procedure may differ between algorithms. Usually hillclimbing is performed in a *beam*search way: at each step the existing model undergoes a modification/addition of a single edge. In order to pick the best edge we must look at $O(N^2)$ possibilities. Since the number of nodes N prohibits us to perform even a linear search at each step, we use *random* hillclimbing in which at each step we choose edges randomly. Specifically, we roll a "3-sided" die with probabilities .8 for addition and .1 for deletion and arc-reversal, and then pick an edge at random to see whether performing the chosen operation improves the global score.

6. Evaluation

The evaluation uses BDeu score described in (Heckerman et al., 1995) and also presented here in equation 3 to compare results between different configurations of our algorithm and to the randomized hillclimbing as described in Section 5.2.

$$S(G,D) = log(\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\frac{1}{q_i})}{\Gamma(\frac{1}{q_i}+N_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\frac{1}{q_i r_i}+N_{ijk})}{\Gamma(\frac{1}{q_i r_i})}) \tag{3}$$

where i is the ith variable, q_i - states of the jth parent of x_i, r - true/false (in our case of binary variables) states of x_i.

The datasets are listed in Table 3. Holdout testsets were used to evaluate overfitting as discussed in Section 6.2.3.

6.1. Datasets

The algorithm has been tested on several real life datasets[1] (sizes shown in Table 3).

Table 3. Data sets and their sizes

Datasets	Entities	Records
Institute	456	1488
Drinks	136	4744
IMDB	100717	49298
Citeseer	104801	180395

1. The *Institute Data* is a set of records of collaborations between professors and students collected from publicly available web pages listed on Carnegie Mellon University Robotic Institute's web site.
2. The *Drinks Data Set* consists of a set of records where each entity is an ingredient in a popular bartending recipe found on the Internet.
3. The *IMDB Data Set* is a collection of casts of actors that participated in movies between the years of 1900 and 1960 extracted from the Internet Movie Database
4. The *Citeseer Data* is a set of co-publication records from the Citeseer online library and index of computer science publications. Since the entities are represented by first initial and last name, a single name might correspond to several people.

[1]We will make the non-proprietory datasets available on the web upon publication

6.2. Empirical Results

We tested our algorithm in a variety of configurations on the datasets listed in Table 3. The results in Table 4 are reported in terms of the average BDeu score, i.e. the final BDeu score obtained by the network averaged over the number of records in the dataset. The number of edges in the resulting Bayes Nets is reported in Table 5. It is interesting to note that the BDeu scores corresponding to the Bayes Nets obtained by running $SBNS$ as described in Table 1 are very close to the ones obtained by random hillclimbing, but have significantly lower number of edges. This supports our claim that the frequent itemsets indeed contain information most relevant to the construction of the highest scoring Bayes Net. It is evident from the results that each of the proposed augmenting algorithms increase the score. We note however that after augmenting the network with highest-mutual-information edges the total number of arcs almost doubled with the highest relative improvement in score when compared to other proposed augmenting techniques. The hillclimbing seems to improve the score even further though the number of edges is almost quadrupled compared to $SBNS$.

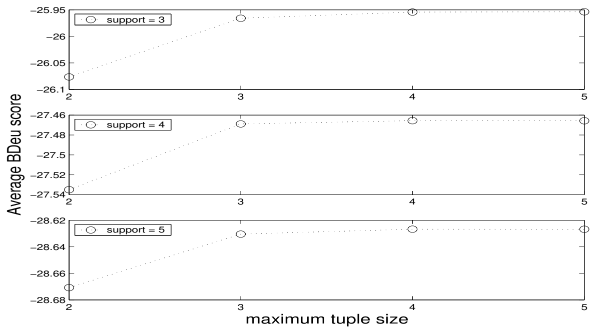

Figure 1. BDeu scores for Citeseer dataset for different parameterizations of the $SBNS$ algorithm

The final score of the DAG produced by $SBNS$ depends on user-defined support and maximum Frequent Set size. We have noticed that for Citeseer, IMDB and Institute datasets lowering support and increasing maximum Frequent Set size results in higher BDeu scores. Figure 1 shows score fluctuations when varying maximum Frequent Set size given fixed support for the Citeseer dataset.

6.2.1. Maximum Frequent Set Size

In our experiments we tried different maximum Frequent Set sizes: ($mfss = 2 \ldots 5$). The lower bound $mfss = 2$ means that we consider only pairs of items and thus the structure learned is based solely on two-way marginal counts. Figure 1 shows that there is an obvious loss in accuracy when high order interactions are not taken into account. Beyond a maximum Frequent Set size of 4 the number of Frequent Sets does not increase substantially in these datasets and hence the behavior of $SBNS$ changes little.

We have to note here, that there is a natural upper bound on the maximum tuple size due to the sparsity of the datasets. For example, there are $94,016$ publications in the Citeseer database that have 2 authors and only $3,022$ that have exactly 6 authors. The potential number of publications that have 6 authors, given the total number of authors in the database is 1.8×10^{27}, so the empirical number is only $(1.6 \times 10^{-22})\%$ of the total. The exponential drop in the number of occurrences as the size of the tuples increases is shown on Figure 2. Hence, we cannot expect a great improvement in the score of the Bayes Net when increasing the maximum tuple size, since there is not enough support for larger tuples.

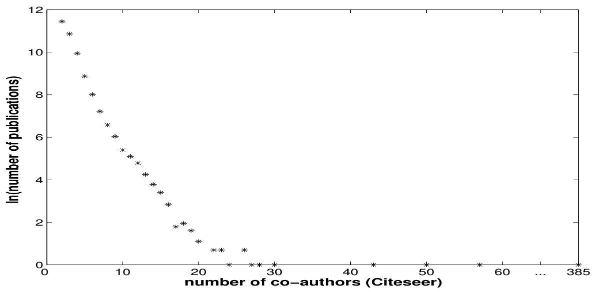

Figure 2. Exponential drop in the number of publications as the number of co-authors increases in the Citeseer Dataset

6.2.2. Support

Lowering support greatly increases the number of Frequent Sets to be considered during screening. However, it also introduces quite a few interactions between variables that have low marginal counts. Model fitting in contingency tables in general is sensitive to very low marginal counts even if they are not zero (Bishop et al., 1977). Here we use BDeu, which is less sensitive to low counts. Despite this, it seems to be a good idea to keep support relatively large. In our case, we have tested a few support sizes on smaller datasets and found $s = 3, 4$ to be reasonable support choices. The overall score of the model seems to be better with $s = 3$, however it seems to overfit more as is shown in Table 6.

6.2.3. Overfitting

We used holdout sets to study overfitting. We withheld roughly a third of the dataset in each case and compared average likelihood per node between the training and testing datasets. The results are summarized in Table 6. The networks learned using $SBNS$ always score higher (better) than those learned by hillclimbing on the testing dataset. This indicates that $SBNS$ algorithm learns better fitting models. As can be seen from Table 6, the difference in average loglikelihood score for training and testing is in

Table 4. Average BDeu scores. ($s = 4$, *mfss* $= 4$; $250,000$ random edges considered for hillclimbing)

dataset	rand hlclmb	$SBNS$	$SBNS$+MIe	$SBNS$+MIe+2nd	$SBNS$+MIe+2nd+hlclmb
citeseer	-33.26	-27.466	-27.375	-27.273	-26.962
imdb	-121.00	-113.15	-112.45	-112.18	-111.28
institute	-11.87	-13.28	-13.18	-13.13	-12.08
drinks	-6.72	-7.21	-7.02	-7.01	-6.705

Table 5. Number of links in the resulting nets. ($s = 4$, *mfss* $= 4$; $100,000$ random edges considered for hillclimbing)

dataset	rand hlclmb	$SBNS$	$SBNS$+MIe	$SBNS$+MIe+2nd	$SBNS$+MIe+2nd+hlclmb
citeseer	88,259	29,004	48,724	53,790	116,558
imdb	112,773	33,434	52,376	57,236	111,281
institute	1,672	346	398	457	1,159
drinks	723	51	123	133	709

general smaller for hillclimbing. Also, the average loglikelihood of the testing set is worse than the training sets, indicating some degree of overfitting. We believe that some overfitting occurs due to the multiple hypothesis testing of hundreds of thousands of possible parents. Correction for multiple hypothesis testing problem (similar to corrections used in other learning algorithms such as (Oates & Jensen, 1998)) will be incorporated into $SBNS$ in the future.

Table 6. Overfitting testing

dataset	train	test
citeseer hillclimb	-30.6738	-31.0127
citeseer $s = 3$	-23.9227	-26.3253
citeseer $s = 4$	-24.1959	-25.0119
imdb hillclimb	-112.81	-114.851
imdb $s = 3$	-98.1607	-110.499
imdb $s = 4$	-100.203	-107.035

6.2.4. Time performance

All experiments were conducted on unloaded 2GHz Pentium IV machines with 2GB of RAM. The total times required to run the algorithm and the time it took random hillclimbing to create a Bayes Net by adding/removing/reversing $250,000$ edges are reported in Table 7.

Table 7. Total times for random hillclimbing, $SBNS$ and $SBNS$+MIe to create a Bayes Net (in mins). ($s = 4$ *mfss* $= 4$)

dataset	rand hillclmb	$SBNS$	$SBNS$+MIe
citeseer	171	59.8	87
imdb	193	225.6	252.8
institute	.53	.016	.017
drinks	.37	.016	.017

We also break the total time into segments corresponding to major steps of the algorithm as reported in Table 8. The biggest cost is to obtain the frequencies; the time it takes to perform the remaining operations depends on the number of Frequent Sets that occur more frequently than predefined support. Our experiments have shown that number to be only a small fraction of the total number of entities (nodes). It is also interesting to note that random hillclimbing is very fast while the network consists of many small subgraphs, but as soon as the subgraphs are joined together by new edges, the time increases tremendously due to the complexity of cycle detection. For example, it takes random hillclimbing on the order of 10 minutes to add/remove/reverse $250,000$ edges, but it takes over 6 hours to perform those operations given the same number of nodes for $300,000$ edges with relatively small increase in the score. In that sense, the random graphs might not be exactly random as discussed in (Callaway et al., 2001).

6.3. Example application

One of the important and growing application fields of large Bayes Nets is recommender systems. A related application is intelligence: having detected a subset of participants of an adverse event, inferring likely accomplices. The purpose of a recommender service is to provide user with suggestion of products that he/she is likely to buy based on their historical preferences. We simulated a recommender query based on the Citeseer dataset. The mapping is as follows: suppose that the set of co-authors of a paper represents user's preferences of particular products. We then learn a Bayes Net based on the available co-authorship information and query the network with incomplete subsets of authors to predict the most likely selection of entities (or authors in our case) that completes the given set.

To answer the query we reported the top n most likely completions with the highest loglikelihoods[2]. Our example query is a subset of former or present members of Daphne Koller's group (DAGS): $\{$*d koller, l getoor, a pfeffer, b taskar*$\}$. Results are presented in Tables 9 and 10.

The suggested completions are in fact people that are either part of or collaborate closely with Daphne Koller's group,

[2]More details, omitted here for space reasons, can be found in (Goldenberg & Moore, 2004)

Table 8. Time (min) per task for $SBNS$. ($s = 4$, $mfss = 4$)

dataset/task	freq sets	local struct search	make Edump & DAG	MI augment	2^{nd} degree augment
citeseer	65.49	4.11	.2	17.2	97.5
imdb	196.22	15.43	13.93	27.23	22.33
institute	.00	.02	.00	.00	.02
drinks	.00	.00	.01	.00	.00

Table 9. Three most likely completions of size 1 for 4 members of DAGS group

completion	score
koller **friedman** pfeffer getoor taskar	-22.523647
koller pfeffer getoor **tong** taskar	-22.694517
koller **boyen** pfeffer getoor taskar	-23.079099

Table 10. Three most likely completions of size 2 for 4 members of the DAGS group

completion	score
koller **grove halpern** pfeffer getoor taskar	-24.065985
koller **malik weber** pfeffer getoor taskar	-24.335174
koller **russell parr** pfeffer getoor taskar	-24.688802

thus by analogy we might expect a set of relevant items to be predicted by the recommender system using this algorithm. It is interesting to note that in the example above the one most likely person to complete the given subset is different (Table 9) than the suggestions provided by the algorithm under the assumption of 2 missing people (Table 10). This observation suggests that there are more complex interactions that could not be found by systems built on pairwise statistics. The inference took less than a second.

7. Related Work

Some of the earlier work in this area has concentrated on efficient representation of sparse data and caching of n-way counts (Moore & Lee, 1998). Chickering and Heckerman (1999) and Meila (1999) have noted that computations requiring one-way and pairwise counts can be sped up significantly when dealing with sparse data using caching and such data structures as ADTrees (Moore & Lee, 1998). We believe that this body of work has great potential and thus we build on the ideas introduced in these papers by utilizing the sparse data representation and low overhead efficient calculation of the marginals.

Using frequent sets when learning Bayes Nets on the local scale was also explored in (Pavlov et al., 2003). The goal of this work was to answer probabilistic queries on a subset of variables, thus there was no need to combine local information to obtain the joint distribution once the query size was estimated. The performance of Bayes Nets learned from a selection of variables was reported to be worse though close in accuracy to the inferences drawn from a Bayes Net learned on a full dataset. In (Hollmen et al., 2003) it has been proposed to integrate Frequent Sets as a local methodology when modelling joint distributions. This work has shown that mixture models obtained from Frequent Sets using maximum entropy are more accurate, thus supporting our claim that frequent sets contain important local information when modelling joint distributions.

One approach to speed up structural search in Bayes Nets for massive datasets has been to restrict the possible parents. The full Sparse Candidate Algorithm is presented in (Friedman et al., 1999). In its original form it is a method to speed up hillclimbing at the cost of lower performance, though in practice the performance loss was shown to be not so great. This work is yet another motivation for us, since structural search on the local scale inadvertently restricts the number of parents. However, since on the global scale the number of parents in our Bayesian Network is not limited we perceive it as an improvement on the original Sparse Candidate algorithm.

Sampling was proposed as one of the techniques to speed up modelling in massive datasets in (Hulten & Domingos, 2002; Pelleg & Moore, 2002). Though an interesting direction it seems to be orthogonal to our approach.

The idea of augmenting Bayes Nets with high mutual information edges is based on the fact that such dependencies could not be accounted for in frequent sets. The fast computation used in this work is based on (Meila, 1999).

8. Conclusion

We have presented a tractable solution to the Bayes Net structure search problem in sparse datasets. Like other researchers, we use Frequent Sets to take advantage of sparseness. Our main new contribution is to perform structural search on the local level in order to produce the global model. We propose several techniques to improve the score of the resulting net. One of the key improvements is augmentation by edges with high mutual information for entities that have not co-occurred in the dataset.

We have performed an empirical study of $SBNS$ using two small and two large (over 10^5 attributes) datasets. We show tractable times while maintaining accuracy better than hillclimbing, which is the only tractable alternative for learning structure in networks of this size.

We believe that SBNS serves two primary purposes. First, it opens new horizons for modelling joint distributions of massive sparse datasets. Second, it can be viewed as a novel way to postprocess Frequent Sets in commercial data mining. We also raise a question of utilizing dataset properties to improve computational complexity of structural search. We feel that there is an immense potential in exploiting other properties such as frequency distribution to obtain high accuracy models in a fraction of time.

9. Appendix

To support our claim that the Frequent Sets contain essential information needed to build a Bayes Net from sparse data, we show that in sparse large datasets positive correlation between two variables is much stronger than negative

Suppose we have 2 binary variables x and y with correlation coefficient ρ. Assume our dataset is sparse and has R records, where R is very large. Under the multinomial sampling model the observed correlation coefficient r is the maximum likelihood estimate of ρ (Bishop et al., 1977).

$$r = \frac{N_{\bar{x}\bar{y}}N_{xy} - N_{\bar{x}y}N_{x\bar{y}}}{\sqrt{N_{\bar{x}+}N_{x+}N_{+\bar{y}}N_{+y}}} \tag{4}$$

Case 1: Two entities have co-occurred v times and kv times separately elsewhere in the dataset, $k \to 0$. Then

$$\begin{aligned} r &= \frac{v(R-2kv-v)-(kv)^2}{\sqrt{(v+kv)^2(R-kv-v)^2}} \\ &= \frac{1}{1+k} - \frac{kv}{R-kv-v} \approx 1, \text{ as } k \to 0 \end{aligned} \tag{5}$$

In fact, only as $k \to \sqrt{\frac{R}{v}} - 1$ (which is clearly a violation of the sparseness assumption), the correlation between x and y becomes significant.

Case 2: Two entities have occurred with frequency Kv but never with each other. K could be rather large, but still conforming to the sparseness assumption, i.e. $Kv \ll R$, then $r = -\frac{1}{\frac{R}{Kv}-1} \approx 0$. Note that when $K = k+1$, we have the same frequency of occurrence as in Case 1, yet only if $K \to \frac{R}{v}$ the correlation would become significant.

We showed that under a sparse assumption, positive correlations are much stronger than negative ones. Thus, when learning a Bayes Net we are much more likely to increase the score by screening Frequent Sets first.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD 12* (pp. 207–216).

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *VLDB 20* (pp. 487–499).

Bishop, Y., Fienberg, S., & Holland, P. (1977). *Discrete multivariate analysis: Theory and practice*. MIT Press.

Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *UAI14*.

Breiger, R. (2003). Emergent themes in social network analysis: Results, challenges, opportunities. *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*.

Callaway, D., Hopcroft, J., Kleinberg, J., Newman, M., & Strogatz, S. (2001). Are randomly grown graphs really random? *Physical Review*.

Chickering, D., & Heckerman, D. (1999). Fast learning from sparse data. *UAI 15*.

Cooper, G., & Herskovits, E. (1991). A Bayesian method for constructing Bayesian belief network from databases. *UAI 7* (pp. 86–94).

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*.

Friedman, N., Nachman, I., & Pe'er, D. (1999). Learning bayes network structure from massive datasets: The "sparse candidate" algorithm. *UAI 15* (p. 206:215).

Goldenberg, A., & Moore, A. (2004). *Tractable structural learning of large bayesian networks from sparse data* (Technical Report CMU-CALD-04-103). CALD, CMU.

Han, J., & Kamber, M. (2000). *Data mining: Concepts and techniques*. Morgan Kaufmann Publishers.

Heckerman, D., Geiger, D., & Chickering, D. (1995). Learning Bayesian Netowrks: The combination of konwledge and statistical data. *Machine Learning*, *20*, 197–243.

Hollmen, J., Seppanen, J., & Mannila, H. (2003). Mixture models and frequent sets: combining global and local methods for 0-1 data. *SIAM ICDM*.

Hulten, G., & Domingos, P. (2002). Mining complex models from arbitrarily large databases in constant time. *ACM SIGKDD 8*.

Mannila, H., & Toivonen, H. (1996). Multiple uses of frequent sets and condensed representations. *KDD 2* (pp. 189 – 194).

Meila, M. (1999). *An accelerated Chow and Liu algorithm: fitting tree distributions to high dimensional sparse data* (Technical Report AIM-1652). MIT.

Moore, A., & Lee, M. S. (1998). Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, *8*, 67–91.

Moreno, J., & Jennings, H. (1938). Statistics of social configuration. *Sociometry*, 342–374.

Oates, T., & Jensen, D. (1998). Large datasets lead to overly complex models: An explanation and a solution. *KDD 4*.

Pavlov, D., Mannila, H., & Smyth, P. (2003). Beyond independence: probabilistic models for query approximation on binary transaction data. *IEEE Transactions on Knowledge and Data Engineering*.

Pelleg, D., & Moore, A. (2002). Using tarjan's red rule for fast dependency tree construction. *NIPS 15*.