Assessing Data Mining Results via Swap Randomization

ARISTIDES GIONIS
Yahoo! Research
HEIKKI MANNILA
University of Helsinki and Helsinki University of Technology
TANELI MIELIKÄINEN
Nokia Research Center
and
PANAYIOTIS TSAPARAS
Search Labs, Microsoft Research

The problem of assessing the significance of data mining results on high-dimensional 0–1 datasets has been studied extensively in the literature. For problems such as mining frequent sets and finding correlations, significance testing can be done by standard statistical tests such as chi-square, or other methods. However, the results of such tests depend only on the specific attributes and not on the dataset as a whole. Moreover, the tests are difficult to apply to sets of patterns or other complex results of data mining algorithms. In this article, we consider a simple randomization technique that deals with this shortcoming. The approach consists of producing random datasets that have the same row and column margins as the given dataset, computing the results of interest on the randomized instances and comparing them to the results on the actual data. This randomization technique can be used to assess the results of many different types of data mining algorithms, such as frequent sets, clustering, and spectral analysis. To generate random datasets with given margins, we use variations of a Markov chain approach which is based on a simple swap operation. We give theoretical results on the efficiency of different randomization methods, and apply the swap randomization method to several well-known datasets. Our results indicate that for some datasets the structure discovered by the data mining algorithms is expected, given the row and column margins of the datasets, while for other datasets the discovered structure conveys information that is not captured by the margin counts.

This work was done while A. Gionis, T. Mielikäinen, and P. Tsaparas were at HIIT, the University of Helsinki.
Authors' addresses: A. Gionis, Yahoo! Research Barcelona, Ocata 1, 1st floor, 08003, Barcelona, Spain; H. Mannila, HIIT Basic Research Unit, University of Helsinki and Helsinki University of Technology, PO Box 68, FIN-00014, Helsinki, Finland; T. Mielikäinen, Nokia Research Center, 955 Pager Mill Rd. Suite 200, Palo Alto, CA 94304-1003; P. Tsaparas (contact author), Search Labs, Microsoft Research, 1065 La Avenida Mountain View, CA 94043; email: tsaparas@cs.helsinki.fi.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms, Management, Experimentation

Additional Key Words and Phrases: Significance testing, randomization tests, 0–1 data, swaps

ACM Reference Format:

Gionis, A., Mannila, H., Mielikäinen, T., and Tsaparas, P. 2007. Assessing data mining results via swap randomization. ACM Trans. Knowl. Discov. Data 1, 3, Article 14 (December 2007), 32 pages. DOI = 10.1145/1297332.1297338 http://doi.acm.org/10.1145/1297332.1297338

1. INTRODUCTION

One of the most important considerations in data mining is deciding whether the discovered patterns or models are significant. While traditional statistics has long been considering the issue of significance testing, it has been given less attention in the data mining community.

In statistics, the methods for significance testing are typically based either on analytical expressions or randomization tests. In this article we focus on the latter approach. We propose the use of *swap randomization* [Cobb and Chen 2003] for assessing data mining results on 0–1 datasets. The basic idea of swap randomization is as follows. Given the dataset D, create random datasets with the same row and column margins as D, run the data mining algorithm on those, and see if the results are significantly different on the real data than on the randomized datasets. If not, then we presume that the results are really due to the row and column margins, and not due to interesting relations in the data. Generating datasets with the same margins as the original is performed by *swaps*, as shown in Figure 1: Take two rows u and v and two columns A and B of the data table with $u(A) = v(B) = 1$ and $u(B) = v(A) = 0$, and change the rows so that $u(B) = v(A) = 1$ and $u(A) = v(B) = 0$. This operation maintains the row and column sums of the dataset, and all datasets with the same row and column sums can be reached through a series of swaps [Ryser 1957; Cobb and Chen 2003].

Swap randomization falls within the broad family of randomization testing methods. Given a metric of interest (e.g., the number of frequent itemsets in the data), randomization testing techniques produce multiple random datasets and test the null hypothesis that the observed metric is likely to occur in the random data. If the metric of interest in the original data deviates significantly from the measurements on the random datasets, then we can reject the null hypothesis and assess the result as significant. The key characteristic of the randomization techniques is in the way that the random datasets are generated. Rather than assuming that the underlying data follows a given distribution and sampling from this distribution, randomization techniques randomly shuffle the given data to produce a random dataset. Shuffling is meant to preserve some of the structural properties of the dataset, for example, in a 0–1 matrix we may want to preserve the total number of 1's in the dataset, or the number of 1's in each column. In the case of swap randomization, the generated samples preserve both the column and row margins. This constraint can also be thought of

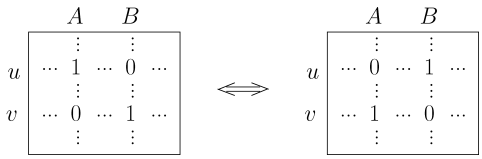

Fig. 1. A swap in a 0–1 matrix.

as a condition on the null hypothesis. We assess the results of a data mining algorithm as significant and interesting if they are highly unlikely to be observed in a random dataset that has the same row and column margins. Using swap randomization we now can answer questions of the following type: Does the observed structure convey any information that is unexpected, given the margins?

Swap randomization is an extension of traditional randomization methods. For instance, a chi-square test for assessing the significance of frequent itemsets is a method based on studying the distribution of datasets where the column margins are fixed, but the row margins are allowed to vary. Similarly, methods that randomize the target value in prediction tasks keep the column margins fixed (e.g., Megiddo and Srikant [1998]), but impose no constraint on the row margins. These techniques are designed for assessing the significance of individual patterns or models, and are not appropriate for assessing complex results of data mining such as clustering or pattern sets. Swap randomization preserves both row and column margins, and takes into account the global structure of the dataset. A motivating example for why it is important to maintain both column and row margins is given in the next section.

Swap randomization has been considered in various applications. An overview is presented in a survey paper by Cobb and Chen [2003]. A very useful discussion on using Markov chain models in statistical inference is Besag [2004], where the case of 0–1 data is used as an example. The problem of creating 0–1 datasets with given row and column margins is of theoretical interest in itself; see, among others Bezáková et al. [2006] and Dyer [2003]. Closely related is the problem of generating contingency tables with fixed margins, which has been studied in statistics (such as Chen et al. [2005]). In general, a large body of research is devoted to randomization methods [Good 2000].

Our contributions in this article are twofold: (i) We describe the algorithmic aspects of swap randomization when applied to large datasets, and (ii) we show how this method can be applied in the data mining setting. In more detail, we give a description of several different ways of generating random matrices with given margins and discuss their performance. Swap randomizations are efficient and can be applied to reasonably large datasets, as our experiments show. We give extensive empirical results showing that some well-known datasets appear to have very few interesting patterns or cluster centers, while other datasets have a lot of structure.

The rest of this work is organized as follows. In Section 2 we present an overview of the swap randomization method, and in Section 3 we discuss the applications of the approach to specific data mining tasks. Section 4 describes

how the random matrices with given margins are generated and provides results on the performance of the algorithms. In Section 5 we describe the experimental results. Section 6 discusses related work, and Section 7 gives some concluding remarks.

2. OVERVIEW OF THE APPROACH

In this section we give an overview of the method. We explain the intuition behind it, describe the algorithmic challenges it poses, and show how it can be applied to testing the significance of results obtained by different kinds of data mining algorithms.

2.1 The Randomization Approach

Let D denote a 0–1 matrix with m rows and n columns that represents our dataset. We view the rows of the matrix as tuples in a database, and the columns as items. The values 0–1 correspond to absence/presence of an item in the tuple. A large range of datasets can be represented in this format.

Assume that we are interested in assessing the result obtained by a particular data mining algorithm $\mathcal{A}$ on input D. Let $\mathcal{A}(D)$ denote the result of the algorithm. For simplicity, assume that it can be described by a single number. For instance, for frequent set mining algorithms, it can be the number of sets whose frequency exceeds a certain support threshold. Similarly, for a clustering algorithm, it can be the error of the clustering solution.

In our randomization approach we generate k datasets $D_1, \ldots, D_k$, such that each $D_t, t = 1, \ldots, k$, is an $m \times n$ 0–1 matrix that has the same row and column sums as the original matrix D. Each dataset D_t is assumed to be a uniform and independent sample from the space of all $m \times n$ 0–1 matrices with the given margins. The algorithm $\mathcal{A}$ is executed on each sampled dataset D_t, yielding results $X_t = \mathcal{A}(D_t)$ for $t = 1, \ldots, k$.

The significance of the result $\mathcal{A}(D)$ of the algorithm $\mathcal{A}$ on the dataset D is tested by comparing it to the set $\mathbf{X} = \{X_1, \ldots, X_k\}$ of the results of $\mathcal{A}$ on the sampled datasets. If the output of the algorithm on the original data does not deviate significantly from the values in $\mathbf{X}$, then the result $\mathcal{A}(D)$ is not surprising and its significance is small; otherwise, the result is considered to be statistically significant.

The statistical significance of $\mathcal{A}(D)$ can be measured in a variety of ways. Assuming that the sampled datasets are independent and that k is large enough so that $\mathbf{X}$ gives an approximation of the real distribution, then the *empirical* p-*value* of $X_0 = \mathcal{A}(D)$ is

$$\frac{1}{k+1}(\min\{|\{t \mid X_t < X_0\}|, |\{t \mid X_t > X_0\}|\} + 1),$$

that is, the fraction of the random datasets in which we see a value more extreme than the value in the real data. The empirical p-value we compute considers both the cases where the value X_0 is very small and very large compared to the values in $\mathbf{X}$. Note that according to our definition the p-value takes values in the interval $[0, 0.5]$.

X	Y
1	1
1	1
1	1
1	1
1	1
1	1
1	1
1	0
1	0
0	1
0	1
0	0
0	0

dataset $\mathcal{D}_1$

X	Y
1	1
1	1
1	1
1	1
1	1
1	1
1	1
1	0
1	0
0	1
0	1
0	0
0	0

dataset $\mathcal{D}_2$

Fig. 2. Examples of two 0–1 datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$. In both cases we are interested in the correlation between columns (attributes) X and Y. The significance of the correlation result might depend on the overall context of the dataset.

Another measure for quantifying the significance of the value X_0 is captured by the Z score

$$Z = \frac{|X_0 - \widehat{X}|}{\widehat{\sigma}},$$

where $\widehat{X} = \mathbf{E}[X_1, \ldots, X_k]$ is the empirical mean of the set $\mathbf{X}$ and $\widehat{\sigma}^2 = \mathbf{Var}[X_1, \ldots, X_k]$ is the empirical variance. Large values of Z indicate that X_0 deviates a lot from the mean of the results obtained on the random datasets. The Z-score can strictly be used to compare the value X_0 against values drawn from a Gaussian distribution. In many of the cases we consider in this article, the distribution of values in $\mathbf{X}$ is clearly not Gaussian. We use the Z-score as a rough indicator, but stress that it should always be used with caution.

2.2 Why Maintain Row and Column Margins?

As mentioned in the Introduction, randomization is widely used as a significance testing method. For example, in control studies in medical genetics it is customary to estimate the interestingness of discovered patterns by a permutation test. In such a test the target variable, namely, the variable describing whether a patient belongs to the case or to the control group, is permuted randomly, and the original data analysis is repeated. The findings on the real data are accepted only if they are stronger than on, say, 99% of the randomized datasets.

However, in many data mining tasks the goal is not to predict a single variable. For example, pattern discovery and clustering look at the structure of the whole dataset. One could, of course, think of randomizing each column of the dataset independently, as implied, for example, in the work of Megiddo and Srikant [1998], but this method ignores some of the structure of the dataset.

As an example, consider the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ in Figure 2. In both datasets variables X and Y are positively correlated, and the itemset $\{X, Y\}$ occurs more often than the independence assumption would imply. As the columns of X and Y are the same for both datasets, any measure of the importance of the association between X and Y that takes only the columns of X and Y

into account will give the same results for $\mathcal{D}_1$ and $\mathcal{D}_2$. However, in dataset $\mathcal{D}_1$, we see that X and Y cooccur in all types of rows, whereas in dataset $\mathcal{D}_2$ the cooccurrence of X and Y happens exclusively in very dense rows. Thus, in $\mathcal{D}_2$ the high frequency of the pair $\{X, Y\}$ is not necessarily due to a correlation between X and Y, but rather to the fact that X and Y tend to occur on rows that have lots of 1's. For example, if X and Y are two items in a market-basket dataset, then their cooccurrence in D_2 would be due to customers that buy almost all items. This finding is not as interesting, as it does not convey information specific to items X and Y.

Consider the datasets $\mathcal{E}_1$ containing 10 copies of $\mathcal{D}_1$, and $\mathcal{E}_2$ containing 10 copies of $\mathcal{D}_2$. The columns for X and Y are the same in both datasets, and in both cases the frequency of the pair is 60. When we generate 1000 random datasets with the margins of $\mathcal{E}_1$, the maximum and average frequencies of $\{X, Y\}$ were 59 and 52.4, and the standard deviation was 2.5. All values were smaller than 60, yielding an empirical p-value of 0.001. For $\mathcal{E}_2$ the corresponding numbers are 69, 63.2, and 2.0. In only 70 cases was the frequency 60 or less, giving an empirical p-value of 0.07. Thus, we can conclude that in $\mathcal{E}_1$ the pair $\{X, Y\}$ is strongly overrepresented, while in $\mathcal{E}_2$ it occurs less often than one would expect. This indicates that the context of the pair of variables has a strong effect on the significance of the frequency of a pair.

The previous example demonstrates the basic concept underlying swap randomization: It takes the bias of row and column counts into account by randomizing over datasets with the same row and column margins as the original dataset. As a result, the notion of interestingness we consider is *conditioned* on the knowledge of the marginal sums. We are interested in assessing information in the dataset that is not conveyed by the marginal sums of the data table. This is the structure that we define as being *interesting*.

As an additional example, consider a dataset whose margin counts satisfy a power-law distribution. The "heavy-tail" property of such a distribution implies that there exist rows and columns in the dataset with very high margin counts. It is possible that on such a dataset, a data mining algorithm discovers patterns that are the direct consequence of the power-law distribution. For instance, columns with high margin counts will tend to form pairs that are much more frequent than pairs of columns with very small margin counts. Alternatively, it is possible that a set of rows and columns with high margin counts is more likely to form clearly distinguished biclusters. Using the framework of swap randomization, one can assess whether a quantity of interest is immediately implied by the power-law distribution on the margin counts, and thus whether it is common to all datasets with the same margin distribution.

In general, a randomization procedure can be viewed as a random process which, given a dataset $\mathcal{D}$, selects a dataset $\mathcal{D}_1$ from some class $\mathcal{C}_D$ of datasets containing $\mathcal{D}$. In swap randomization, the class $\mathcal{C}_D$ is the set of all datasets having the same row and column margins as $\mathcal{D}$. Varying the class $\mathcal{C}_D$ defines a different randomization approach. In the independent permutation approach, the class $\mathcal{C}_D$ contains all datasets with the same column sums as $\mathcal{D}$. The row sums are allowed to vary, so we expect the distribution to be fairly uniform. As a consequence, the two methods differ significantly. A data mining result may

be deemed significant by the independent permutation method, but still be explained purely by the row and column margins. We will see such an example in the empirical section.

Maintaining row and column margins assumes that both row and column sums are in some way interesting quantities. For example, consider ecological presence/absence data of species and locations, where rows to species and columns to locations. Then the column sums correspond to the commonness of species, and row sums correspond to the species diversity of locations: Both quantities are of fundamental interest in ecology, and hence maintaining them exactly makes sense. Similarly, consider a document dataset where rows correspond to documents and columns to words. Then row counts measure the number of different words in the document, and column sums indicate how widely used is the word. In this case it is clear that the distributions of the row and column sums are important properties of the dataset: In many cases they explain a lot of the observed structure of the data. Generating random datasets that maintain these sums makes it possible to see whether some structure is unexpected, given the margins.

There are other techniques of randomization. For example, instead of requiring that the row and column margins are maintained exactly, we could require that they are maintained approximately. One possible condition is to require that the L_1-distance between the row margins of $\mathcal{D}$ and $\mathcal{D}_1$ is bounded. Another randomization procedure could be obtained by requiring that the *distribution* of row and column margins is approximately the same in $\mathcal{D}$ and $\mathcal{D}_1$.

2.3 Limitations of the Approach

Summarizing the previous section, we believe that comparison of datasets with the same margins provides a good method for assessing the significance of the discovered patterns. However, what constitutes an interesting pattern is not well defined and often can be very subjective. Therefore, we do not claim that swap randomization is a panacea for assessing all potential notions of interestingness. In fact, through the framework of swap randomization the concept of interestingness takes a very precise definition: Interesting patterns are those that are not likely to appear in datasets with the same margins as the input dataset. Having a precise definition of interestingness is certainly a desirable property, but it is possible that there are many other intuitive concepts of interestingness that are not captured by this definition.

Consider a binary dataset in which half of the rows contain only 1's and the other half only 0's. This particular dataset is the *unique* dataset with those exact margins, and, according to the swap randomization framework, no pattern will be considered interesting even though the dataset has very interesting structure. The point here is that we must remember that swap randomization does nothing more than compare datasets with the same margins. In this particular example, the technique reports that there is no interesting structure, *given* the margins of the dataset. In fact, one can argue that all the interesting structure of the dataset is expected given the margins, since this is the only dataset with these margins.

Another limitation of the swap randomization framework, as presented in this article, is that it is applied only to data with 0–1 values, which denote absence or presence of an attribute value. Many datasets can be modeled as 0–1 matrices; examples include market-basket data, word occurrences in documents and access logs, data from scientific applications such as biology, ecology, paleontology, and more. However, many other datasets require more general representations. Two important cases widely studied in the literature are numerical data and categorical data.

For numerical data one possible extension is generate datasets in which the sums of the entries of all rows and all columns are fixed. This is the problem of sampling from the space of contingency tables and it has been studied extensively in the statistics literature [Chen et al. 2005; Cobb and Chen 2003; Diaconis and Gangolli 1995; Snijders 1991]. One basic technique is to sample by performing a random walk on the *Diaconis chain* [Diaconis and Saloff-Coste 1995], which is a generalization of the basic swap move shown in Figure 1. A move on the Diaconis chain consists of randomly selecting two rows i and j and two columns k and l, and adding the values 1, -1, -1, and 1 at the entries (i,k), (i,l), (j,k), and (j,l) (respectively) of the current matrix, as long as no entry becomes negative.

In order to apply the framework to numerical data, in addition to the technical modifications, it must be verified that it is meaningful to maintain row and column sums. For instance, it arguably makes sense to maintain row and column sums in a dataset in which the presence or absence information of words in documents is replaced with counts. On the other hand, it is clearly not meaningful to maintain row and column sums in a dataset with the numerical attributes `age`, `income`, and `num_of_children`. Similarly, the swap randomization framework does not seem to be applicable to data with categorical attributes, even if these attributes have a binary domain. Consider a dataset with attributes `gender`, `education`, and `employment_status`. It is not clear how to perform the basic swap moves, or what are the margins we wish to maintain.

2.4 Generating Matrices with Given Margins

Generating random 0–1 datasets with given row and column sums is a technical challenge. This problem has been studied extensively in statistics [Chen et al. 2005; Cobb and Chen 2003], theoretical computer science [Bezáková et al. 2006; Dyer 2003], and in various application areas [Kashtan et al. 2004; Milo et al. 2002].

In this article we use a *Markov chain* approach to the problem of sampling. Starting from the original dataset, we make a small random local move which interchanges a pair of 1's with a pair of 0's and does not change the row and column sums, thus producing a new dataset with the same margins. Such a local move is called a *swap*, and a sequence of swaps is performed until the data is sufficiently mixed to provide a random sample. In the swap randomization framework we want to compare the input dataset with random datasets that have the same margins. Since we have no reason to assume that any of these datasets is more likely than the others, we would like to sample *uniformly* from

the set of all possible datasets. Uniform sampling avoids introducing any sampling bias to the significance test. In the language of Markov chains we want to ensure that the stationary distribution of the chain is the uniform distribution.

The state space of the Markov chain consists of all datasets with the given margins. There is a transition between two datasets if there is a swap that transforms one to the other. The Markov chain is reversible, that is, a swap can be undone by a single (reverse) swap. However, the chain is not regular, as some datasets (states) have more neighbors than others. This implies that the stationary distribution of the chain is not the uniform distribution. Therefore, a straightforward application of swapping does not guarantee uniform sampling.

The problem of nonuniformity can be fixed in at least two ways: (i) by using the *Metropolis-Hastings* algorithm [Hastings 1970; Metropolis et al. 1953], which is a well-studied method for converting one Markov chain with stationary distribution π to another with stationary distribution π'; and (ii) by adding multiple *self-loops* in order to guarantee that all states have the same degree.

For applying the Metropolis-Hastings algorithm, one needs to compute the degree of any given state of the chain, that is, the number of all valid swaps for a given 0–1 matrix. We give a simple formula for computing the degree at each state, and we show how to maintain this quantity incrementally. The complexity of incremental maintenance of the state degree is $O(\min\{m, n\})$ for an $m \times n$ matrix, making the algorithm somehow inefficient. On the other hand, adding self-loops does not require computing any additional expensive information; so while more steps are needed for convergence, the time complexity of each step is, in expectation, constant, making it a very efficient algorithm in practice.

3. USING THE FRAMEWORK

In this section we describe how the swap randomization framework can be applied to different data mining tasks, such as finding frequent sets and correlations, clustering, and spectral analysis. Our methodology allows us to investigate the significance of the patterns that exist in a given dataset, at different levels of granularity.

First, we are able to characterize the significance of global aspects of the dataset. If the number of frequent sets, or the number of highly correlated pairs contained in the dataset, is not significant with respect to that found in a randomly rearranged dataset, then we can conclude that the dataset does not contain any interesting global structure of frequent sets, or of highly significant correlations.

Additionally, we can also look at individual itemsets. In this case we are interested in identifying itemsets whose frequency is smaller or larger in the sampled datasets when compared with the original dataset. If the frequency of an itemset drops in the sampled dataset, it is implied that the frequency cannot be explained by the margins of the dataset. If the frequency increases, a possible explanation is that the items in the itemset are anticorrelated in the original dataset.

The aforesaid observations also apply when mining simple association rules. Recall that the accuracy (confidence) of a rule $(X \Rightarrow B)$ is defined as $\frac{f(XB)}{f(X)}$,

where $f(XB)$ and $f(X)$ are the frequencies of $X \cup \{B\}$ and $\{X\}$, respectively. Assume now that X is a singleton set. Since $f(X)$ remains fixed, the confidence of the rule is proportional to the frequency $f(XB)$. Therefore, the significance of the rule $(X \Rightarrow B)$ is determined by the significance of the pair $\{X, B\}$. In the general case that the set $\{X\}$ consists of more than one item, our observation that the confidence of $(X \Rightarrow B)$ depends only on the frequencies of $f(XB)$ and $f(X)$ still holds. Thus, we can assess the significance of the rule $(X \Rightarrow B)$ by keeping track of the frequencies of the sets $X \cup \{B\}$ and $\{X\}$ on the sample datasets that we generate during the swap randomization process. Overall, for the framework of swap randomization, the case of association rules is subsumed by the analysis of frequent itemsets.

Swap randomization can be applied to testing the significance of clustering results. Given a clustering algorithm like k-means and a target number of clusters k, we compare the clustering error in the original dataset with that in the sampled datasets. If the difference is large, then we can deduce that the dataset has meaningful cluster structure. This simple approach turns out to yield very clear results on synthetic datasets with known cluster structure, as we will see in Section 5.

To obtain some intuition of how clustering structure is related to row and column margins, consider the very simple and rather contrived dataset $\mathcal{D}$ which has n rows, n columns, and k "pure" clusters. Here, by pure clusters we mean that the data matrix is block diagonal with k blocks of 1's, each of size $\frac{n}{k} \times \frac{n}{k}$, and the rest of the matrix entries are 0's. Therefore all the row and column margins are $\frac{n}{k}$, and there is a perfect clustering into k clusters with error 0. On the other hand, it is true that a random dataset with the same margins as $\mathcal{D}$ has no clustered structure; each entry of the data matrix of such a random dataset appears to be almost random with probability $\frac{1}{k}$, and the error of clustering into k clusters is roughly $O(\frac{1}{k}(1 - \frac{1}{k})n^2)$.

A different notion of global structure is captured in the singular values and vectors of the data matrix. Singular vectors capture linear trends in the dataset. The corresponding singular values capture the strength of the linear trend, that is, the tendency of rows or columns to align with corresponding singular vectors. The strongest linear trends can be used to construct a low-dimensional approximation of the dataset with provable approximation error. In randomly generated data, the strongest linear trends should be determined by the marginals of the dataset. This is usually the first singular value. The remaining dataset has no structure; thus we expect the remaining singular values to be small. If the original data contains some linear structure, then the top singular values (especially the nonprincipal ones) should be higher than those of random datasets with the same margins.

4. SAMPLING DATASETS WITH GIVEN ROW AND COLUMN MARGINS

4.1 Basics

We now describe the process of sampling a matrix from the space of all $m \times n$ 0–1 matrices with given margins.

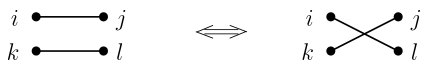

Fig. 3. A swap in the graph representation G_D.

Let D be a 0–1 dataset with m rows and n columns. We denote by r_i the sum of the ith row of D, $i = 1, \ldots, m$, and by c_j the sum of the jth column, $j = 1, \ldots, n$. An equivalent way to represent the input matrix D is as a bipartite graph $G_D = (R, C, E)$, with $|R| = m$ and $|C| = n$. Vertex $i \in R$ corresponds to the ith row of D, vertex $j \in C$ to the jth column of D, and $(i, j) \in E$ if and only if $D(i, j) = 1$ for all i and j. The degrees of the vertices of the graph are r_i for $i \in R$, and c_j for $j \in C$.

The main idea is to start from the graph G_D corresponding to the original dataset and to perform a *local swap* that leaves the margins unchanged. When many such swaps have been performed, the resulting graph can be considered as a random dataset drawn randomly from the stationary distribution.

In more detail, a local swap in a bipartite graph $G = (R, C, E)$ can be defined by four vertices, i, j, k, and l of G, such that $i, k \in R$ and $j, l \in C$, and $(i, j) \in E$, $(k, l) \in E$, $(i, l) \notin E$, $(k, j) \notin E$. A new dataset $G' = (R, C, E')$ is then formed by updating the edges of $G = (R, C, E)$ as follows.

$$E' \leftarrow E \setminus \{(i, j), (k, l)\} \cup \{(i, l), (k, j)\}$$

In other words, we remove the current edges $\{(i, j), (k, l)\}$ and add new edges $\{(i, l), (k, j)\}$. Visually, a local swap is depicted in Figure 3 for the graph representation and in Figure 1 for the matrix representation.

Formally, a local swap is a step on a Markov chain $\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$, where the state space $\mathcal{S}$ is the set of all graphs with the given degree sequences, and $\mathcal{T}$ is the set of transitions defined by swaps. In other words, the set $\mathcal{T}$ contains all pairs of graphs (G, G') such that it is possible to obtain G' from G (or vice versa) by performing a single swap.

4.2 Naïve Nonuniform Approach

Algorithm 1 shows a straightforward implementation of this Markov approach.

Algorithm 1. `Naïve`

Input: Graph G_D, number of random walk steps k_n
Output: Graph G with the same degree sequences as G_D
1: $G \leftarrow G_D$
2: **while** $k_n > 0$ **do**
3: $G' \leftarrow$ `Find_adjacent` (G)
4: $G \leftarrow G'$
5: $k_n \leftarrow k_n - 1$
6: **end while**
7: Return G

Finding the next transition $(G, G') \in \mathcal{T}$ from graph G, that is, executing line 3 of algorithm `Naïve`, is not a completely straightforward task. The simplest way is to pick a pair of edges in G, reject if the edges are not swappable, and repeat until a pair of swappable edges is found. This is shown in Algorithm 2. Alternatively, one could store all swappable pairs in a structure and select one uniformly at random. The selection process becomes faster, but there is additional cost of updating the data structure at each step.

Algorithm 2. `Find_adjacent`

Input: Graph G
Output: Graph G' that differs from G in exactly one swap (i.e., $(G, G') \in \mathcal{T}$)
  repeat
    Select edges $(i, j), (k, l) \in E(G)$ uniformly at random
  until $(i, l) \notin E(G)$ and $(k, j) \notin E(G)$
  $E(G') \leftarrow E(G) \setminus \{(i, j), (k, l)\} \cup \{(i, l), (k, j)\}$

Given graph G, the algorithm `Find_adjacent` generates a graph G' uniformly at random among all graphs G' such that $(G, G') \in \mathcal{T}$. The reason is that there exists an one-to-one correspondence between the set of such graphs G' and the set of swappable pairs of edges: Each graph G' is mapped to exactly one unique swappable pair, and for each swappable pair there exists a unique graph G'. Algorithm `Find_adjacent` clearly samples uniformly at random from the set of swappable pairs: Each swappable pair is sampled with probability proportional to $2/|E|^2$.

Now, in order for the Markov chain to sample graphs uniformly at random from the set $\mathcal{S}$, the following conditions have to hold:

(1) The state space $\mathcal{S}$ is connected under the transitions of $\mathcal{M}$.
(2) The Markov chain $\mathcal{M}$ has a uniform stationary distribution.
(3) Starting from G_D, a sufficiently large number of local swaps should be performed so that the chain *mixes*. We would like to know how many such swaps should be performed, namely, the mixing time of the chain.

Connectedness. The Markov chain is connected. One can move from any state of the chain to any other state using swaps [Ryser 1957; Cobb and Chen 2003].

Uniformity. First notice that the Markov chain $\mathcal{M}$ is reversible. Now, for each graph (state) $G \in \mathcal{S}$, we define $d(G)$, namely the *degree* of the Markov chain $\mathcal{M}$ at G, to be the number of different graphs (states) G' such that $(G, G') \in \mathcal{T}$. From the theory of Markov chains, it is well known that the stationary distribution of a reversible chain is proportional to the degree at each state in the underlying transition graph. Therefore, in order to obtain a uniform distribution, all states of the Markov chain must have the same degree. A simple construction shows that this is not true in general for the Markov chain $\mathcal{M}$. Therefore, the `Naïve` algorithm (Algorithm 1) does not converge to the uniform distribution.

Mixing time. The mixing time of the Markov chain we defined previously has been the object of theoretical study [Cobb and Chen 2003], but without any conclusive results. It is estimated that running the chain for a number of steps in order of the number of 1's in the matrix is sufficient for convergence. We do not deal with theoretical aspects of convergence, but study it empirically in the experimental section. See Besag [2004] for a discussion on convergence and p-values.

4.3 The Self-Loop Method

Straightforward application of the Markov chain approach does not produce uniform sampling. There are two ways to fix this bias and obtain uniform distribution. The first is by adding self-loops, as shown in Algorithm 3. Algorithm `Self_loop` works as `Naïve` does. It samples pairs of edges until it finds a swappable pair. Its difference from `Naïve`, however, is that in `Self_loop` all steps are counted and decrease the counter; thus nonswappable pairs of edges are counted as self-loops. The reason that `Self_loop` leads to uniform distribution is that when self-loops are counted, the degree of each $G \in \mathcal{S}$ becomes fixed and equal to $|E|^2$. Each pair of edges, swappable or nonswappable, contributes one to the degree of all states.

Algorithm 3. `Self_loop`

Input: Graph G_D, number of random walk steps k_s
Output: Graph G_D, number of random walk steps k_s
1: $G \leftarrow G_D$
2: **while** $k_s > 0$ **do**
3: Select edges $(i, j), (k, l) \in E(G)$
4: **if** $((i, l) \notin E(G)$ and $(k, j) \notin E(G))$ **then**
5: $E(G') \leftarrow E(G) \setminus \{(i, j), (k, l)\} \cup \{(i, l), (k, j)\}$
6: **end if**
7: $k_s \leftarrow k_s - 1$
8: **end while**
9: Return G

4.4 The Metropolis-Hastings Approach

The second way of sampling from the uniform distribution is by using the Metropolis-Hastings algorithm [Hastings 1970; Metropolis et al. 1953], which is a standard method for converting a Markov chain with stationary distribution π to another Markov chain with stationary distribution π'. In our case, $\pi(G) \sim d(G)$ and we want $\pi'(G) \sim 1$, so the Metropolis algorithm becomes as shown in Algorithm 4. For some swap that takes the algorithm from state (graph) G to state G', if the state G' has higher degree, then the algorithm performs the swap with probability $\frac{d(G)}{d(G')}$. The algorithm assumes knowledge of the degree $d(G)$ for each graph $G \in \mathcal{S}$. We will discuss soon how $d(G)$ can be computed.

Algorithm 4. `Metropolis-Hastings`

Input: Graph G_D, number of random walk steps k_m
Output: Graph G with the same degree sequences as G_D
1: $G \leftarrow G_D$
2: **while** $k_m > 0$ **do**
3: $G' \leftarrow$ `Find_adjacent`(G)
4: $G \leftarrow G'$, with probability $\min\{1, \frac{d(G)}{d(G')}\}$
5: $k_m \leftarrow k_m - 1$
6: **end while**
7: Return G

4.5 Running Time

We now analyze the running time of the algorithms. We will prove some results on the complexity of the approaches, including a result that characterizes the degree of a state in the Markov chain. The conclusion in this section is that the `Self_loop` algorithm is always more efficient than the `Metropolis-Hastings`.

First, we assume that we can sample edges in constant time and that we can test whether a pair of edges is swappable in constant time. The former task can be performed by keeping all edges in an array, while the latter can be performed by keeping in memory the data D in the matrix form, or by storing all edges in a hash table.

The running time of `Find_adjacent` is a random variable and depends on the number of swappable edges for each graph (state) G. Recall that the number of swappable pairs of graph G is $d(G)$. Therefore, the probability of finding a swappable pair of edges is precisely $\frac{d(G)}{|E|^2}$, thus the expected time for staying in G is $\frac{|E|^2}{d(G)}$. Without counting the self-loops, the probability of visiting graph G is $\frac{d(G)}{2|\mathcal{T}|}$, which is precisely the stationary distribution of algorithm `Naïve` at G. Thus, the expected running time of algorithm `Find_adjacent` is

$$T_F = \sum_{G \in \mathcal{S}} \frac{|E|^2}{d(G)} \cdot \frac{d(G)}{2|\mathcal{T}|} = \frac{|E|^2}{2} \cdot \frac{|\mathcal{S}|}{|\mathcal{T}|}. \tag{1}$$

Notice that $|\mathcal{T}|/|\mathcal{S}| = O(|E|^2)$, since the degree of each graph G in $\mathcal{S}$ is at most $|E|^2$. On the other hand, the following lemma is immediate.

LEMMA 4.1. *For bipartite graphs $G = (U, V, E)$ in which the maximum degree is $o(|E|)$, we have $|\mathcal{T}|/|\mathcal{S}| = \Omega(|E|^2)$.*

PROOF. Notice that the random walk leaves the degrees at each vertex unaffected in all states. Given any state (graph G) in $\mathcal{S}$, consider an edge $(i, j) \in E(G)$. Any other edge $(k, l) \in E(G)$ can be swapped with (i, j) unless either $l \in \Gamma(i)$ or $k \in \Gamma(j)$ (or both), where $\Gamma(i)$ are the neighbors of i in the bipartite graph. Thus, the number of edges that should be excluded from swapping with (i, j) is $o(|E|)$, yielding a total number of at least $(|E| - o(|E|)) \cdot |E|$ swappable pairs. Since each state in $\mathcal{S}$ has degree $\Omega(|E|^2)$, the lemma follows. □

COROLLARY 4.2. *For bipartite graphs $G = (U, V, E)$ whose degree distribution follows a power law with $\alpha > 2$ we have $|\mathcal{T}|/|\mathcal{S}| = \Omega(|E|^2)$.*

PROOF. For simplicity assume that $|U| = |V| = n$. For power laws with exponent $\alpha > 2$ we have $|E| = O(n)$ in expectation and the maximum degree is $n^{\frac{1}{\alpha-1}} = o(n)$ (e.g., see Newman [2003]). Thus, the conditions of Lemma 4.1 are satisfied. □

The preceding results imply that for some important classes of datasets (such as graphs with bounded degrees or degrees that follow a power-law distribution) the expected time T_F of the `Find_adjacent` algorithm is constant. Thus, for those classes of data, the running time of algorithm `Naïve` is $T_N = T_F \cdot k_n = O(k_n)$. Similarly, for the `Self_loop` algorithm the overall running time is $T_S = O(k_s)$. Furthermore, the expected time spent in each state for performing self-loops (before moving out to a new state) is constant.

We now turn to the running time of `Metropolis-Hastings`. This running time can be written as $T_M = T_D^0 + k_m(T_F + T_D)$, where T_F is the running time of `Find_adjacent`, T_D is the time needed to compute $d(G')$ given that $d(G)$ is already computed, and T_D^0 is the time needed to compute $d(G)$ for the first time. Next we explain how to compute $d(G)$ and how to update the computation for $d(G')$. The time needed for the update is linear with respect to $\min\{m, n\}$.

THEOREM 4.3. *Let $G = (U, V, E)$ be a bipartite graph represented as a binary matrix D with $m = |U|$ rows and $n = |V|$ columns. Let r_i be the "left" degree of node $i \in U$, c_j be the "right" degree of node $j \in V$, and define $M = DD^T$. Then, the number of graphs G' that are yielded from G with one local swap is equal to*

$$d(G) = J(G) - Z(G) + 2K_{22}(G), \tag{2}$$

where

$$J(G) = \frac{1}{2}\left(|E|(|E|+1) - \sum_{i \in U} r_i^2 - \sum_{j \in V} c_j^2\right) \tag{3}$$

is the number of disjoint pairs of edges,

$$Z(G) = \sum_{(i,j) \in E} (r_i - 1)(c_j - 1) \tag{4}$$

is the number of "Z" structures

$$\{(a, b), (c, d), (c, b) \in E, \text{ with } a, b, c, d \text{ all distinct}\},$$

and

$$K_{22}(G) = \sum_{\substack{i,k \in U \\ i \neq k}} \binom{M(i,k)}{2} = \frac{1}{2} \sum_{\substack{i,k \in U \\ i \neq k}} M(i,k)^2 - M(i,k) \tag{5}$$

is the number of $K_{2,2}$ cliques of G.

PROOF. Eq. (2) follows from the observation that every disjoint pair of edges is a swappable pair unless it is part of a Z structure. However, the value $J(G) - Z(G)$ underestimates the number of swappable pairs: In each $K_{2,2}$ there are 2

disjoint pairs of edges and 4 Z's, since each pair participates in two Z structures. Therefore, we need to add 2 for each $K_{2,2}$ to obtain the correct count, resulting in Eq. (2).

We now explain how to derive Eqs. (3)–(5). In order to compute the number of pairs of disjoint edges, consider a single edge (i, j). This edge forms a disjoint pair with all other edges $(|E|-1)$, except those that have an endpoint at i (r_i-1) or j (c_j-1), that is, with $|E|+1-r_i-c_j$ other edges. Therefore, summing over all edges and dividing by two to avoid double-counting, we obtain

$$\begin{aligned} J(G) &= \frac{1}{2}\sum_{(i,j)\in E}(|E|+1-r_i-c_j) \\ &= \frac{1}{2}\left(|E|(|E|+1)-\sum_{i\in U}\sum_{j\in V:(i,j)\in E} r_i-\sum_{j\in V}\sum_{i\in U:(i,j)\in E} c_j\right) \\ &= \frac{1}{2}\left(|E|(|E|+1)-\sum_{i\in U} r_i^2-\sum_{j\in V} c_j^2\right). \end{aligned}$$

For the computation of the number of Z structures, it suffices, to observe that there is a one-to-one mapping between a Z structure and the edge in the middle of the Z. A given edge (i, j) creates $(r_i-1)(c_j-1)$ Z structures with the edges incident to i and j. Summing over all edges we obtain Eq. (4).

For the number of $K_{2,2}$'s we observe that the value of the entry $M(i,k)$ is the cardinality of the set $V_{ik} = \{j : (i, j) \in E \wedge (k, j) \in E\}$. Any pair of elements in the set V_{ik} forms a $K_{2,2}$ with nodes i, k. There are $\binom{M(i,k)}{2}$ such pairs. Summing over all pairs $i, k \in U$ we obtain Eq. (5). □

COROLLARY 4.4. *Given graphs G and G' such that $(G, G') \in \mathcal{T}$, $d(G')$ can be calculated from $d(G)$ in time $O(\min\{m, n\})$.*

PROOF. Without loss of generality assume that $\min\{m, n\} = m$, and that we are using the $m \times m$ matrix $M = DD^T$. Otherwise we can use the $n \times n$ matrix $M' = D^T D$. Using Eq. (2) we have

$$d(G') = d(G) - \Delta Z + 2\,\Delta K_{22}.$$

Graphs G and G' differ only by one swap; so, matrices $D(G)$ and $D(G')$ differ only in four positions, and matrices $M(G)$ and $M(G')$ differ only in two rows and two columns. Therefore ΔZ can be computed in constant time and ΔK_{22} in $O(m)$ time. □

We note that `Metropolis-Hastings` still needs to run the `Find_adjacent` algorithm for finding a candidate swap pair. Although the way that the algorithm moves between states is different (neighboring states are not chosen uniformly at random), and thus it may guarantee faster convergence, we believe that most likely this does not offset the additional cost incurred by the computation of the degrees. Thus we prefer to experiment with the `Self_loop` algorithm. We note that it may be possible to maintain the number of swappable pairs in linear time, thus eliminating the cost of the `Find_adjacent` algorithm. However, this is a nontrivial task, and still does not guarantee that

Table I. The Datasets

Dataset	# of rows	# of cols	# of 1's	dens. (%)
ABSTRACTS	128820	25335	10449902	0.32
ABSTRACTS′	128803	5918	7150992	0.94
COURSES	2405	5021	65152	0.54
KOSARAK	990002	41270	8019015	0.02
PALEO	124	139	1978	11.48
RETAIL	88162	16470	908576	0.06

the `Metropolis-Hastings` algorithm would be faster. Recall also that in many cases, the cost of `Find_adjacent` is constant in expectation.

5. EMPIRICAL RESULTS

We perform experiments with many of the well-known datasets used in the data mining community. A description of the datasets we are using is as follows: ABSTRACTS contains document-word information on a collection of project abstracts submitted for funding by NSF. ABSTRACTS′ is a pruned version of ABSTRACTS, where we keep only words of medium frequency (with frequency between 200 and 8854). COURSES is a student-course dataset of courses completed by computer science students of the University of Helsinki. RETAIL is a market-basket dataset collected in a Belgian supermarket [Brijs et al. 1999]. KOSARAK is a click-stream dataset from a Hungarian news website. Finally, PALEO, the smallest dataset, contains information of species fossils found in specific palaeontological sites in Europe [Fortelius 2006]. Exact information of the datasets, including sizes and the density of 1's, are shown in Table I.

Swap randomization maintains the row and column margins of the datasets. Figure 4 shows the distribution of the row and column sums for datasets KOSARAK, RETAIL, and PALEO. We see typical power-law shapes for KOSARAK. For RETAIL, the column sums seem to have the power law, while the row sums have slightly different shapes. In PALEO the rows (fossil sites) and columns (species) have been selected so that very small row or column counts do not appear.

5.1 Convergence and Performance

We have tested extensively the convergence properties of the swapping Markov chain for various datasets. Designing diagnostics for the convergence of a Markov chain is an open research question, so our tests can provide evidence only that the chain is mixed and by no means do they constitute a proof. See Besag [2004] and Besag and Clifford [1991, 1989] for detailed discussions on how p-values can be obtained from this type of chain.

An example is shown in Figure 5. For each of our datasets, we measure the number of frequent itemsets for a given threshold. The y-axis in Figure 5 shows the number of frequent itemsets in the sampled datasets, divided by the number of frequent itemsets in the original dataset. The x-axis shows the number of steps in the Markov chain scaled by the number of 1's in the corresponding dataset, namely, position $x = i$ shows a sample after iL steps, where L is the number of 1's in the corresponding dataset. We see that in almost all cases the

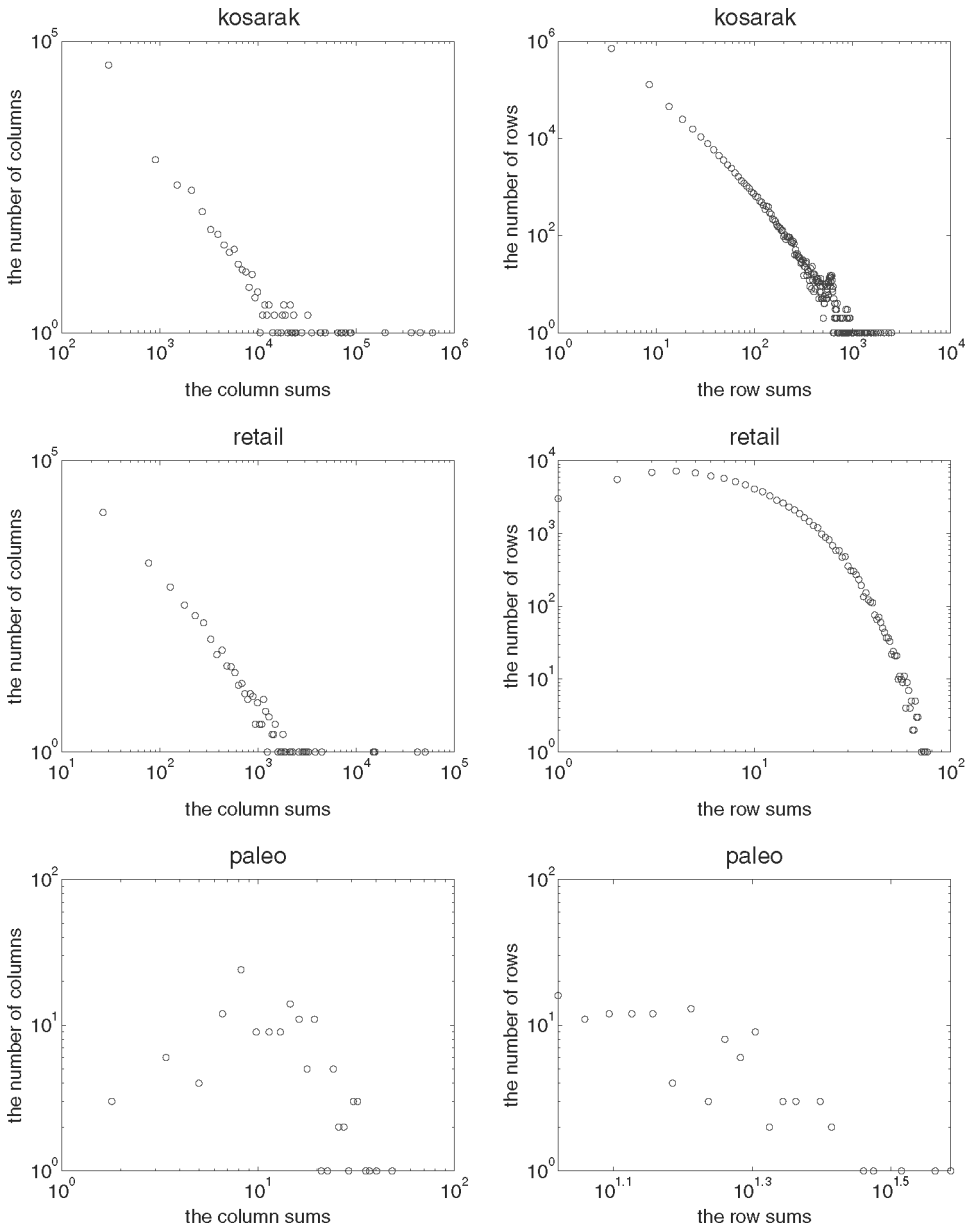

Fig. 4. Distribution of row and column sums in KOSARAK, RETAIL, and PALEO datasets. Log-log scale.

chain mixes quite rapidly: Already after L steps ($4L$ in the case of KOSARAK) the number of frequent sets has stabilized.

Similar convergence evidence was obtained for all our measures: frequencies of specific itemsets, number of correlations above a certain threshold, clustering errors, etc. In all of the experiments presented in the following sections we have run the chain with a very large number of steps in order to ensure convergence.

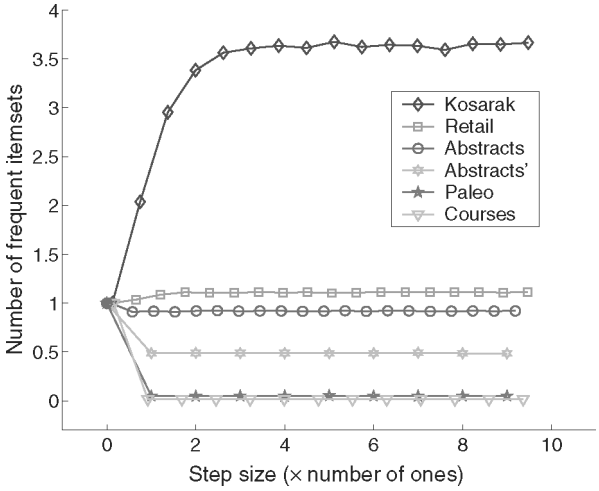

Fig. 5. Convergence: The x-axis is the number of steps ($\times$ the number of 1's in the data); the y-axis is the number of frequent itemsets in the sampled datasets, divided by the number of frequent itemsets in the original dataset.

Table II. Running Times Needed to Perform Swap Randomization on the Different Datasets

Dataset	time	Dataset	time
ABSTRACTS	12m53s	KOSARAK	8m38s
ABSTRACTS′	9m11s	PALEO	0.100s
COURSES	3.35s	RETAIL	1m1.5s

We report the clock time (in s) needed to perform a number of swaps equal to 5 times the number of the 1's in the dataset.

Additionally, the swaps can be performed quite efficiently. Table II shows the running time for the different datasets, using a simple Perl implementation on a 3GHz Pentium machine with 2GB of memory. The reported times are for performing $5L$ swaps. In most cases a much smaller number of swaps can be used. For comparison, the time needed to cluster the COURSES and PALEO datasets into 5 clusters by using a simple implementation of the K-means algorithm are 60 s and 1 s, respectively. Thus, the time needed for swapping is small compared to the need for running the data mining algorithm on the samples.

5.2 Frequent Itemsets

In this section we assess the results of frequent itemset mining using swap randomization. Table III shows the number of frequent sets for the datasets described in Section 5. We compute the collections of frequent sets in the original data, in random data under swap randomization, and in random data under independent permutations of columns (i.e., only column margins are maintained). The collections are denoted by $\mathcal{F}$, $\mathcal{F}_s$, and $\mathcal{F}_p$, respectively. The minimum support thresholds were chosen so that the number of frequent sets is not exceedingly large. Frequent items, that is, frequent sets of size 1, are

Table III. The Number of Frequent Itemsets

ABSTRACTS, $minsupp = 5000$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	1128	1004.8	4.8	698.6	3.7
≥ 3	226	188.7	2.5	75.6	2.0
ABSTRACTS′, $minsupp = 600$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	4854	839.5	19.2	22.5	4.0
≥ 3	223	0.0	0.0	0.0	0.0
COURSES, $minsupp = 400$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	9687	442.2	12.5	149.1	2.9
≥ 3	9412	259.7	11.4	46.8	2.3
≥ 4	8479	62.8	5.9	1.2	0.5
≥ 5	6669	3.1	1.3	0.0	0.0
KOSARAK, $minsupp = 5000$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	1436	5644.5	60.8	266.0	1.3
≥ 3	977	5013.8	59.5	88.8	0.9
≥ 4	417	3629.6	52.5	7.1	0.3
≥ 5	95	1864.6	35.2	0.0	0.0
≥ 6	8	589.2	16.0	0.0	0.0
PALEO, $minsupp = 7$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	2828	266.7	14.8	227.7	11.7
≥ 3	2058	9.8	5.4	4.9	3.3
≥ 4	898	0.0	0.2	0.0	0.0
RETAIL, $minsupp = 200$					
$\|X\|$	$\|\mathcal{F}\|$	mean($\|\mathcal{F}_s\|$)	std($\|\mathcal{F}_s\|$)	mean($\|\mathcal{F}_p\|$)	std($\|\mathcal{F}_p\|$)
≥ 2	1384	1616.1	12.3	860.3	6.8
≥ 3	489	569.0	9.1	168.2	3.2
≥ 4	78	79.2	3.7	7.6	1.0

$|X|$: the minimum cardinality of the itemset we include to the count. $|\mathcal{F}|$: the number of frequent sets of cardinality at least $|X|$ in the original data; $|\mathcal{F}_s|$: the expected number of frequent sets in swapped data and its standard deviation; $|\mathcal{F}_p|$: the expected number of frequent sets in random data with the same column margins as the original data and its standard deviation. The expectations and standard deviations were computed on 500 experiments.

omitted from the table since they do not change by swapping or permuting the columns. In the case of swapped and permuted datasets, we show the mean values (and standard deviations) of 500 randomized versions of the datasets.

Table III clearly demonstrates the differences between the randomization methods. All datasets seem to contain many interesting frequent itemsets when compared to the corresponding datasets with permuted columns. The sizes of the frequent set collections are always considerably smaller in the permuted data than in the original data. On the other hand, different datasets show very different behaviors with respect to swapping. In ABSTRACTS and RETAIL the number of frequent sets remains about the same under swap randomization, whereas in ABSTRACTS′, COURSES, and PALEO the numbers decrease significantly. Finally, there is a considerable increase in the number of frequent sets in KOSARAK.

Interpreting the results, we can conclude that the structure captured by frequent itemsets in ABSTRACTS and RETAIL can be attributed mainly to the row and column margins, and thus is preserved in random datasets where the margins are preserved. On the other hand, in the datasets ABSTRACTS′, COURSES, and PALEO, the structure captured by frequent sets is more interesting, since it disappears under swap randomization. Note that with respect to frequent itemsets, the main difference between datasets ABSTRACTS and ABSTRACTS′ is the elimination of very frequent words from ABSTRACTS. Thus any frequent set structure in ABSTRACTS is mostly due to the stop words. Associations between stop words (e.g., frequently occurring pairs such as "of the," "a," and "of") capture mostly the syntactic structure in the document rather than the semantic one. They are typically considered as less interesting, since they are expected to occur in most documents that share the same syntactic structure. This is accurately identified by swap randomization.

The increase in the number of frequent sets in the case of KOSARAK implies that many sets of items are anticorrelated with each other. A possible explanation for this phenomenon lies in the structure of the data. KOSARAK consists of anonymized click-stream data from a news portal: The link structure of the websites can cause negative correlations between groups of pages.

Table IV shows in more detail what happens to the number of frequent pairs in KOSARAK, RETAIL, and PALEO datasets under swap randomizations and permutations of the columns, for different values of the frequency threshold. We observe for the PALEO dataset that the number of frequent pairs is higher in the original dataset than in the swapped data for moderate and large values of the frequency threshold. This indicates that cooccurrences of variables are more common in the original data. When the frequency threshold is very small, there are more frequent pairs in the swapped data. The reason for this is that in the limit, almost every pair tends to occur in the swapped (or permuted) dataset.

For the KOSARAK and RETAIL datasets the results cannot be given for arbitrarily small frequencies because the task of finding frequent pairs becomes prohibitively expensive. For moderate values of the threshold we observe in both cases that swapped data has more frequent pairs; the ratios stay remarkably similar for different thresholds. Only for the largest threshold in KOSARAK do we obtain a frequent pair that is more frequent than in the swapped data. This behavior can probably be explained by the existence of disjoint dense blocks of 1's in the data; we have conducted simple experiments and simulations showing that in such datasets the behavior of the number of frequent pairs has such a trend. We continue the discussion on the behavior of pair frequencies in Section 5.3.

Although the number of frequent itemsets is indicative of the structure that is contained in the data, it is not informative with respect to what the actual itemsets contained in the collections are, and how the collections relate to each other. It may well be the case that collections have about the same size, yet are completely disjoint. In the following we present some results on how the itemset collections change under swap randomization. Our results are meant to be indicative of the behavior of the swap randomization, and to help guide

Table IV. Number of Frequent Sets of Size 2 for KOSARAK, RETAIL, and PALEO Datasets for Different Values of Frequency Threshold

KOSARAK dataset					
	original	swapped		permuted	
minsupp	# of 2-sets	mean	std	mean	std
1500	4003	5322.7	24.2	1087.5	4.1
1750	3083	3939.5	13.5	850.2	3.9
2000	2423	3136.9	11.2	702.4	3.1
2500	1567	2118.3	8.2	508.5	2.5
3000	1161	1541.7	6.9	385.6	1.7
4000	660	934.3	3.7	243.9	1.5
5000	459	630.7	2.9	177.3	0.9
10000	139	195.9	1.2	74.7	0.5
20000	45	61.2	0.4	29.0	0.1
30000	28	37.0	0.1	19.0	0.1
40000	17	25.9	0.4	12.0	0.1
50000	14	18.7	0.5	8.0	0.0
100000	4	6.0	0.0	4.0	0.0
300000	1	0.0	0.0	0.0	0.0
RETAIL dataset					
	original	swapped		permuted	
minsupp	# of 2-sets	mean	std	mean	std
200	895	1046.7	7.1	692.1	5.6
250	640	722.0	5.4	467.1	4.7
300	471	517.3	4.8	342.7	3.7
400	284	311.5	3.2	208.5	2.7
500	191	208.8	2.3	142.2	2.2
600	137	151.9	1.8	92.8	1.5
800	71	80.3	1.9	50.8	1.1
1000	49	51.4	1.0	35.9	1.0
2000	20	17.6	0.5	13.0	0.0
3000	10	11.0	0.2	7.0	0.0
4000	7	7.0	0.0	7.0	0.0
PALEO dataset					
	original	swapped		permuted	
minsupp	# of 2-sets	mean	std	mean	std
1	4262	6884.4	44.2	6669.1	40.7
2	3092	4198.7	32.7	3939.4	39.2
3	2394	2403.0	24.7	2197.6	32.9
4	1799	1354.5	22.3	1211.6	24.1
5	1384	768.0	19.1	674.9	18.1
6	1017	439.7	16.1	383.5	14.8
7	770	257.0	12.3	223.3	10.3
8	577	151.9	8.9	131.7	8.1
9	425	91.7	7.1	78.8	6.5
10	289	55.9	5.6	47.1	5.2
15	39	4.7	1.9	3.6	1.6
20	11	0.2	0.4	0.1	0.4

Minsupp: the frequency threshold; original: the number of frequent pairs in the original data; swapped: the average number of frequent pairs in swapped datasets in its standard deviation; permuted: the average number of frequent pairs in permuted datasets and its standard devitation. The expectations and standard deviations were computed on 200 experiments.

Table V. Changes in the Collections of Frequent Sets

Dataset	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{F}_s\rvert$	std	$\frac{\lvert\mathcal{F}\cap\mathcal{F}_s\rvert}{\lvert\mathcal{F}\rvert}$	$\frac{\lvert\mathcal{F}\setminus\mathcal{F}_s\rvert}{\lvert\mathcal{F}\rvert}$
ABSTRACTS	1128	1004.8	4.8	0.767	0.233
ABSTRACTS′	4854	839.5	19.2	0.083	0.917
COURSES	9687	442.2	12.5	0.042	0.958
KOSARAK	1436	5644.5	60.8	0.724	0.276
PALEO	2828	266.7	14.8	0.045	0.955
RETAIL	1384	1616.1	12.3	0.882	0.118

D: the dataset; $\mathcal{F}$: the frequent itemset collection in the dataset; $\mathcal{F}_s$: the frequent itemset collection in the swapped dataset; std: the standard deviation in the size of the frequent itemset collection in the swapped dataset; $\frac{\lvert\mathcal{F}\cap\mathcal{F}_s\rvert}{\lvert\mathcal{F}\rvert}$: the fraction of itemsets that are preserved in the collection; $\frac{\lvert\mathcal{F}\setminus\mathcal{F}_s\rvert}{\lvert\mathcal{F}\rvert}$: the fraction of frequent itemsets that disappear from the collection. The values involving swapped data are expectations on 500 experiments.

future analysis. A formal statistical analysis of the statistical significance of individual itemsets is beyond the scope of this article.

Table V shows the average fraction of itemsets in the original dataset that are preserved or disappear after swap randomization. For the datasets ABSTRACTS, KOSARAK, and RETAIL where the size of the collection remains relatively stable (or in the case of KOSARAK increases), the mean fraction of preserved itemsets is around 70%, confirming our intuition that the original collection did not contain much interesting structure. This is especially pronounced in the case of the RETAIL data, where on average 88% of the frequent itemsets are preserved. For the remaining datasets, the mean fraction of preserved itemsets drops below 9%.

The swap randomization method can be used also to suggest unexpected itemsets in the data, namely, sets that are very frequent in the original data but very rare in the swapped data. For example, {`dissertation, doctoral, promising`} is common in the ABSTRACTS′ data (support 682), but rare in the corresponding swapped data (mean support 2.4). Similarly, the set {`differential, equations, partial, solutions`} has support 679 in the original data, while its mean support in swapped data is less than 0.4. The most "dull" itemset is the set {`address, result`}, with supports 691 and 691.6 in the original and swapped data, respectively.

Figure 6 shows how the frequencies of individual itemsets of size 3 change in the ABSTRACTS′ datasets under swap randomizations. We see that all the frequencies decrease by at least a factor of 10, and that there are fairly large differences in the decrease. Some sets with original frequency of about 650 have frequency less than 10 in the swapped version, while others have a frequency of more than 60. This means that the average column and row sums for rows in which these sets occur have to be quite different. The differences also suggest that the ratio $f(X)/f_s(X)$, where $f(X)$ is the frequency of X in the original data and $f_s(X)$ the frequency in the swapped data, might be a useful descriptor of the itemset.

5.3 Correlations

We now study how correlations between items (columns) change under swap randomization. We compute correlation between two columns using the Pearson

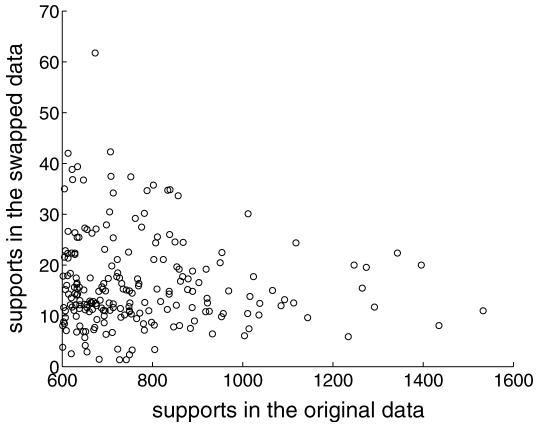

Fig. 6. Frequencies of itemsets of size 3 in ABSTRACTS′ dataset and its swapped versions.

correlation coefficient, which returns a value between -1 (perfectly anticorrelated) and 1 (perfectly correlated). Computing all pairwise correlations between the columns in the data tables is computationally expensive for most of our datasets. Instead, we focus on the k columns with the highest column degree, for $k = 100$. The rationale is that items appearing frequently are usually more interesting and we want to study their correlations. Furthermore, this allows for an interesting comparison with the randomization technique that permutes columns independently. Since the column counts are large, our experiments give an indication as to how the row counts affect the significance of the pair.

Table VI and Figure 7 show our results for different datasets. We compute the maximum and minimum correlation values, as well as the number of pairs whose correlation exceeds a certain threshold, for different thresholds. We present the values for the original data, as well as the mean value, and the Z-score for both swapped and independently permuted data. The statistics are taken over 100 different samples.

From the results we make the following observations. As expected, when randomizing the dataset, strong correlations, either positive or negative, tend to disappear for both methods of randomization. However, the way in which this happens differs between the two methods. For the independent permutation model, the correlations concentrate very sharply and almost symmetrically around zero. For swap randomization, negative correlations disappear at a much faster rate, for example, for RETAIL and ABSTRACTS they disappear almost completely. On the other hand, the number of positive correlations remains relatively high, indicating that to some extent the correlations in the dataset (especially low ones) can be explained by the row and column margins. This becomes especially clear when one compares the mean values for the swap and independent model in the RETAIL and ABSTRACTS datasets. On the other hand, for the COURSES and ABSTRACTS′ datasets we observe that positive correlations drop significantly faster.

Figure 7 shows the histograms of the correlations for three of the datasets: KOSARAK and RETAIL, which had somewhat unintuitive results for the number

Table VI. Statistics for Correlation Values

		Swapped		Permuted	
Measure	Original	mean	Z	mean	Z
		ABSTRACTS Dataset			
max	0.47	0.06	11.19	0.01	514.2
min	−0.10	−0.01	11.90	−0.01	137.5
≥0.03	1941	667.67	7.28	0.00	—
≥0.02	2649	3573.88	5.22	0.00	—
≥0.01	3363	4904.27	7.91	0.86	3342
≤−0.01	776	6.41	21.19	0.66	775
≤−0.03	310	0.77	23.58	0.00	—
		ABSTRACTS′ Dataset			
max	0.51	0.03	14.89	0.01	592.5
min	−0.05	−0.00	15.15	−0.01	54.58
≥0.03	760	5.76	18.35	0.00	—
≥0.02	1391	37.32	11.10	0.00	—
≥0.01	2379	3455.31	6.50	1.20	2342.6
≤−0.005	1033	5.20	22.97	174.49	63.3
≤−0.01	691	1.92	37.24	0.59	844.4
		COURSES Dataset			
max	0.91	0.24	57.19	0.08	114.38
min	−0.53	−0.03	75.25	−0.07	77.05
≥0.30	565	0.00	—	0.00	—
≥0.10	2025	1611.06	10.86	0.01	20209.9
≥0.03	2923	3214.10	6.55	365.9	149.9
≤−0.01	1373	20.23	244.29	1574.1	5.8
≤−0.03	1058	1.18	886.30	332.4	40.7
		RETAIL Dataset			
max	0.40	0.11	87.16	0.01	303.1
min	−0.02	−0.01	18.32	−0.01	11.46
≥0.03	219	113.83	15.04	0.00	—
≥0.02	537	480.17	4.02	0.00	—
≥0.01	1513	2100.27	14.32	15.92	352.73
≤−0.005	458	2.65	257.85	307.3	9.24
≤−0.01	92	0.00	—	1.88	65.3
		PALEO Dataset			
max	0.87	0.41	10.77	0.15	14.6
min	−0.42	−0.24	7.98	−0.05	29.55
≥0.20	1011	145.00	70.16	0.13	632.8
≥0.10	1430	839.81	24.44	2.79	475.1
≥0.03	1756	1968.30	6.37	275.26	95.5
≥0.01	1841	2319.17	13.51	1084.38	33.6
≤−0.01	2984	2159.15	22.20	1195.8	4.56
≤−0.10	2204	593.15	80.70	0.00	—

A row of type max contains the value of the largest correlation, while a row of type, say ≥0.01, contains the number of correlation pairs with value greater than 0.01. The empirical p statistic in all the given results is 1%.

of frequent pairs, and PALEO, a nicely behaving dataset (as a comparison). We observe that permutation of columns gives nicely symmetric distributions of the correlations; this, of course, is no surprise. Swapping the KOSARAK dataset results in some negative correlations, while for the retail dataset the shape of the histogram spreads somewhat. Figure 7 is an example of the power of

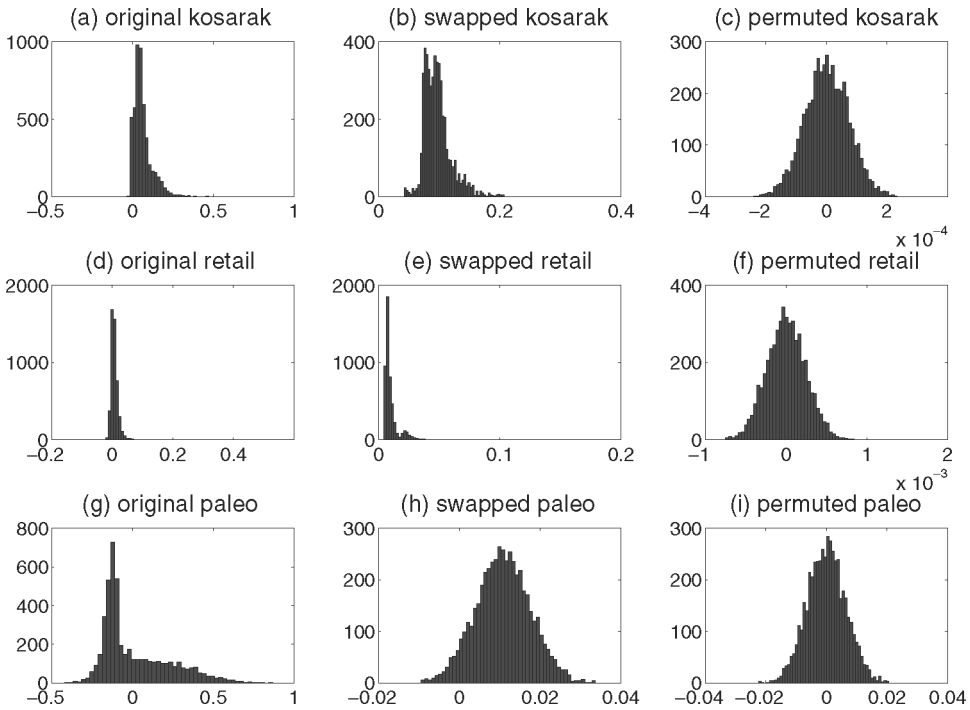

Fig. 7. Distribution of correlations in KOSARAK, RETAIL, and PALEO datasets; the 100 most frequent variables from each dataset: (a) original KOSARAK; (b) swapped KOSARAK; (c) permuted KOSARAK; (d) original RETAIL; (e) swapped RETAIL; (f) permuted RETAIL; (g) original PALEO; (h) swapped PALEO; (i) permuted PALEO. For each pair of variables the correlation was computed in 100 replications of swapping and permutation, and the swapped and permuted panels show the histograms of the averages over the replications.

randomization tests: panel (b), for example, seems as if it would indicate interesting structure in the dataset. However, this structure can be explained by the row and column margins of the data in the sense that random datasets with the same margins have similar structure.

5.4 Clustering

Our results on assessing the clustering structure of datasets are shown in Table VII. We perform our clustering experiments using Matlab's k-means default function. We obtain results only for datasets of small and medium sizes; our largest datasets cannot be clustered by Matlab's k-mean function. For each dataset, we measure the clustering error for the original dataset for clustering with $k = 10$ and $k = 20$, which is shown in the third column of Table VII. The clustering error is measured as the sum of square of distances from each point to its representative cluster center, and is computed by the default Matlab k-means function. Then we sample 100 sampled datasets, which we cluster with the same parameters. We compute the mean and standard deviation of the clustering error in the sampled datasets, which are shown in the fourth and fifth columns of Table VII. The sixth column (Z) reports the distance in standard

Table VII. Results on Clustering

Dataset	k	E	mean	std	Z	p
S1	10	1777.3	3669.9	11.1	170.43	0.01
	20	1660.7	3303.2	11.3	145.33	0.01
S2	10	4075.4	4084.4	11.6	0.77	0.22
	20	3686.2	3691.3	12.1	0.42	0.36
COURSES	10	17541.6	24405.1	30.2	227.09	0.01
	20	16062.0	23588.4	31.9	235.92	0.01
PALEO	10	1040.7	1401.7	4.8	74.74	0.01
	20	800.1	1193.9	5.9	67.09	0.01
RETAIL	10	23920.9	24086.0	135.2	1.22	0.10
	20	22276.3	22481.1	235.3	0.87	0.18

k: number of clusters used in k-means; E: clustering error in the original dataset; mean: mean clustering error in the sampled datasets; std: standard deviation of the clustering error in the sampled datasets; Z: distance of E from mean, measured in standard deviations; p: empirical p-value.

deviations between the error in the original dataset and the mean error in sampled datasets. Finally, the last column records the empirical p-value as described in Section 2.1.

In addition to experiments on real datasets, we also verify the method on synthetic ones, which we generate as follows: We first generate k cluster centers to be d-dimensional binary vectors, where each entry is 0 or 1 with probability $\frac{1}{2}$. Then for each cluster center we generate $\frac{n}{k}$ points, where for each point we first copy all the entries of its cluster center and then each entry is inverted with probability e. For both of the datasets S1 and S2 shown in Table VII we use $n = 1000$, $d = 20$, and $k = 10$. For S1 we use $e = 0.1$, while for S2 we use $e = 0.45$. In other words, the clustered structure of S2 is hugely corrupted with noise. We see that our results indicate that S1 has indeed a clustered structure, while this is not the case for S2. Notice that even though both datasets have a ground truth of 10 clusters, we obtain similar results for running the k-means algorithm with $k = 20$, that is, when the ground-truth number of clusters is not known. For the real datasets, we see again that all datasets have clustered structure except for RETAIL.

5.5 Singular Values

We compute the top-20 singular values for the randomized sets, and compare the average value of each singular value with the corresponding one of the original dataset. We observed that in most cases, the first singular value of the random datasets is relatively large, compared to the first singular value of the original data. The explanation is that the first singular value captures the linear trend that is defined by the degree sequences. In contrast, the nonprincipal singular values are significantly smaller in random datasets. Thus, we conclude that swap randomization destroys linear trends in the data.

Since the sum of squares of the singular values is equal in the original and swapped data, there should be a "crossing point" when plotting the singular values of both datasets in decreasing order (see Figure 8). This actually suggests an interesting heuristic for estimating the correct number of dimensions when

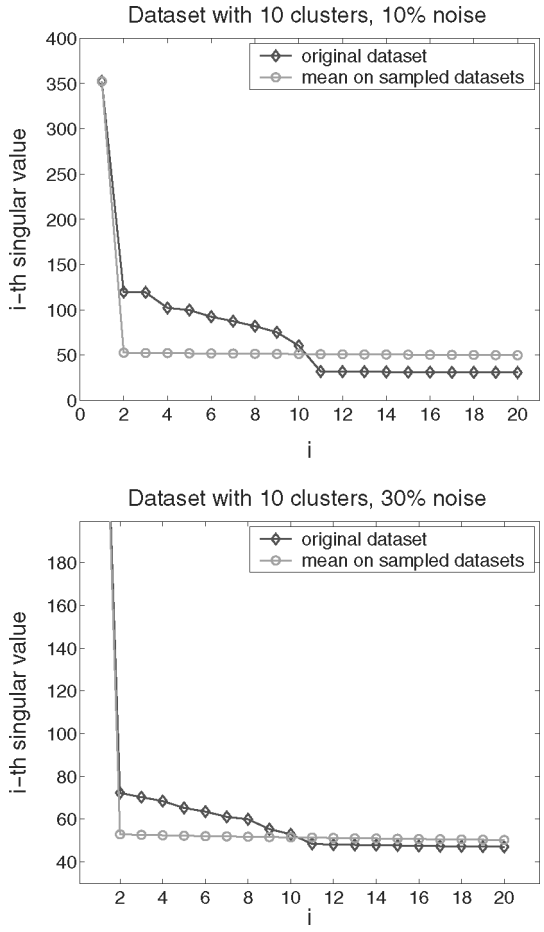

Fig. 8. Singular values of the original and sampled datasets (mean out of 100 samples) for synthetic datasets. The crossing point of the two lines at position 10 corresponds to the number of clusters planted in the data.

projecting to low-dimensional spaces. The index of the first singular value that is significantly lower than the corresponding random one is probably a good indicator that the structure contained in the remaining singular vectors is no more interesting than that contained in a random matrix.

Finding a good low-dimensional approximation of the data is an important problem in many aspects. First, it provides a concise description of our data. Second, as already mentioned in Section 3, principal singular vectors capture the strongest linear trends in the data. Each vector defines a direction along which the data points are aligned. Since the vectors are orthogonal, projecting along these directions usually results in a natural clustering of the data points, and thus each singular vector corresponds to a different cluster. One important problem in this setting is to identify the number of dimensions on which to project. We want to eliminate vectors that capture noise, and maintain only the ones that model the actual structure in the dataset.

We observe that in many cases the crossing point in the singular-values plot can help guide this decision. For example, the PALEO data is conjectured to contain three clusters and the crossing point for this data is indeed at position 3. We experiment further with the aforementioned idea on synthetic data in which we can plant a known number of clusters. The data was generated with the procedure described in the previous section. Note that in the case where no noise is added in the data generation, the number of singular vectors is equal to the number of planted clusters. The addition of noise does not add structure, so we expect the new singular values to be insignificant according to our measure. Figure 8 shows indeed that for a dataset with 10 clusters, the crossing point is at position 10, for noise levels ranging from 10% to as high as 30%.

6. RELATED WORK AND ADDITIONAL COMMENTS

Defining the significance of discovered patterns has attracted a lot of attention in data mining. In one of the first known papers, Brin et al. [1997] considered measuring the significance of associations via the chi-square test. A lot of other measures have been proposed to capture the interestingness of patterns, such as DuMouchel and Pregibon [2001], Jaroszewicz and Simovici [2001], Liu et al. [2001, 1999], Megiddo and Srikant [1998], and Xiong et al. [2004]. Megiddo and Srikant [1998] use ideas from statistical significance testing to estimate the number of false discoveries and rank the set of the association rules found. Liu et al. [2001] present an algorithm to identify "nonactionable" association rules, that is, rules that are significant but covered by other, more specific rules. DeMouchel and Pregibon [2001] apply statistical filtering for association rules using a Bayesian framework. Recently, Webb [2007, 2006] built upon some of the existing approaches and presented two generic approaches for discovering significant rules while filtering false discoveries. A comprehensive presentation and comparison of the methods that capture interestingness of patterns can be found in Tan et al. [2002].

The problem is also very well studied in statistics, and there is a significant amount of work for sampling from the space of contingency tables [Chen et al. 2005; Cobb and Chen 2003; Diaconis and Gangolli 1995; Snijders 1991], as well as several studies that give asymptotics on the exact number of such tables, such as Wang and Zhang [1998]. The algorithmic properties of some of these methods are discussed in Bezáková et al. [2006]. A good survey on the topic is provided by Chen et al. [2005].

In theoretical computer science the subject has drawn attention in the context of providing bounds for the mixing of the Markov chain. Very recently, Bezáková et al. [2006] gave a polynomial-time algorithm for sampling binary 0–1 matrices with given margins. The algorithm is based on a different Markov chain than that based on swaps.

The problem of generating random matrices with fixed margins has also been studied in many application areas, such as ecology [Sanderson 2000] and biology [Kashtan et al. 2004], and analysis of complex networks [Milo et al. 2002].

In using the swap randomization framework, $\mathcal{D}_i$ is obtained from $\mathcal{D}$ by making a sequence of swaps. As a single swap has only very limited effect on the

dataset, one might look for implementations of $\mathcal{A}$ that take advantage of the incremental characteristics of the datasets for which $\mathcal{A}$ is being used. This is fairly straightforward, for example, for frequent sets, but we have not pursued this idea further.

Finally, we remark that it is possible to directly generate random datasets that do not preserve exact row and column sums but on expectation. This involves setting each entry (i, j) equal to 1 with probability $r_i c_j / L$, and equal to 0 otherwise. This expectation model has the drawback that the fraction $r_i c_j / L$ has probability interpretation only if $\max\{r_i, c_j\} \leq L$. The advantage of this model is that the generation process has fixed complexity (quadratic), which is likely faster than the Metropolis-Hasting algorithm. Furthermore, it is amenable to formal analysis. Experimenting with this model, we found that it gives similar results as the swap method, but sometimes is slightly inaccurate. For instance, such inaccuracies were observed in the KOSARAK dataset in which both the row and column sums follow power-law distribution. Additionally, the savings in running time are not significant.

7. CONCLUSIONS

We have studied the algorithmic properties of swap randomization, and described how it can be used in assessing results of data mining algorithms. We gave an algorithmic treatment of the swap randomization method, showing some results on the computation of the number of immediately reachable states in the Markov chain, and we showed that the `Self_loop` algorithm is always more efficient than the `Metropolis-Hastings` method for this problem. Our work shows that swap randomization is efficient in practice, and that it can be used for large datasets.

We have conducted extensive experiments on the use of swap randomization. The results show large differences in the amount of structure that is present in different datasets. For example, when viewed through clustering or frequent sets, the RETAIL dataset apparently has very little structure, apart from its very skewed degree distribution for columns and (slightly less) for rows. The number of frequent sets in the real dataset is about the same as in the randomized versions, and clustering the original and randomized versions yields about the same error. On the other hand, several of the other datasets clearly have lots of second-order structure, as evidenced by the dramatic drop in number of frequent sets and strong correlations when moving to the randomized version of the data.

Swap randomization is a technique that maintains the first-order statistics of the data. Thus, it should not be used to study the significance of discoveries that depend only on the first-order statistics of the data, namely, the row and column margins; power laws are an example of these types of statistics. An interesting question is whether it is possible to generate, from a dataset D, random datasets having the same margins as D while keeping also some second-order statistics (e.g., the frequency of certain variable pairs) fixed. It can be shown that one can indeed efficiently generate datasets that have the given row and column margins, and additionally maintain counts $\{f(X_i)\}$, where the sets X_i are pairs

and the collection $\{X_i\}$ is acyclic. Maintaining frequencies of sets in general is NP-hard [Mielikäinen 2003; Calders 2004].

There are also other types of statistics that might be interesting to maintain in 0–1 data. We have already remarked on the possibility of maintaining the counts approximately. Another option would be to maintain the counts of small subgraphs, such as cliques $K_{2,2}$ or $K_{2,3}$ [Tomkins 2006].

Availability. Software for swap randomization can be found at `http://www.cs.helsinki.fi/hiit_bru/software/swaps/`.

ACKNOWLEDGMENTS

We thank Jean-François Boulicaut, Céline Robardet, and Jérémy Besson for interesting discussions that got us started on swaps, and Dimitris Achlioptas and Andrew Tomkins for useful remarks.

REFERENCES

Besag, J. 2004. Markov chain Monte Carlo methods for statistical inference. http://www.ims.nus.edu.sg/Programs/mcmc/files/besag_tl.pdf.

Besag, J. and Clifford, P. 1989. Generalized Monte Carlo significance tests. *Biometrika 76*, 4, 633–642.

Besag, J. and Clifford, P. 1991. Sequential Monte Carlo p-values. *Biometrika 78*, 2, 301–304.

Bezáková, I., Bhatnagar, N., and Vigoda, E. 2006. Sampling binary contingency tables with a greedy start. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SIAM, 414–423.

Bezáková, I., Sinclair, A., Stefankovic, D., and Vigoda, E. 2006. Negative examples for sequential importance sampling of binary contingency tables. http://arxiv.org/abs/math.ST/0606650.

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. 1999. Using association rules for product assortment decisions: A case study. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 254–260.

Brin, S., Motwani, R., and Silverstein, C. 1997. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, 265–276.

Calders, T. 2004. Computational complexity of itemset frequency satisfiability. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 143–154.

Chen, Y., Diaconis, P., Holmes, S. P., and Liu, J. S. 2005. Sequential Monte Carlo methods for statistical analysis of tables. *J. Amer. Statis. Assoc. 100*, 469, 109–120.

Cobb, G. W. and Chen, Y.-P. 2003. An application of Markov chain Monte Carlo to community ecology. *Amer. Math. Month. 110*, 264–288.

Diaconis, P. and Gangolli, A. 1995. Rectangular arrays with fixed margins. In *Discrete Probability and Algorithms*, 15–41.

Diaconis, P. and Saloff-Coste, L. 1995. Random walk on contingency tables with fixed row and column sums. Tech. Rep., Department of Mathematics, Harvard University.

DuMouchel, W. and Pregibon, D. 2001. Empirical Bayes screening for multi-item associations. In *Knowledge Discovery and Data Mining*, 67–76.

Dyer, M. 2003. Approximate counting by dynamic programming. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, San Diego, CA, 693–699.

Fortelius, M. 2006. Neogene of the old world database of fossil mammals (NOW). http://www.helsinki.fi/science/now/.

Good, P. 2000. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*, 1, 97–109.

JAROSZEWICZ, S. AND SIMOVICI, D. A. 2001. A general measure of rule interestingness. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 253–265.

KASHTAN, N., ITZKOVITZ, S., MILO, R., AND ALON, U. 2004. Efficient sampling algorithm for estimating dubgraph concentrations and detecting network motifs. *Bioinf. 20*, 11, 1746–1758.

LIU, B., HSU, W., AND MA, Y. 1999. Pruning and summarizing the discovered associations. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 125–134.

LIU, B., HSU, W., AND MA, Y. 2001. Identifying non-actionable association rules. In *Knowledge Discovery and Data Mining*, 329–334.

MEGIDDO, N. AND SRIKANT, R. 1998. Discovering predictive association rules. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, 274–278.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A., AND TELLER, E. 1953. Equations of state calculations by fast computing machines. *J. Chem. Phys. 21*, 1087–1092.

MIELIKÄINEN, T. 2003. On inverse frequent set mining. In *Proceedings of the 2nd Workshop on Privacy Preserving Data Mining (PPDM)*, IEEE Computer Society, 18–23.

MILO, R., SHEN-ORR, S., ITZKOVIRZ, S., KASHTAN, N., CHKLOVSKII, D., AND ALON, U. 2002. Network motifs: Simple building blocks of complex networks. *Sci. 298*, 824–827.

NEWMAN, M. 2003. The structure and function of complex networks. *SIAM Rev. 45*, 2, 167–256.

RYSER, H. J. 1957. Combinatorial properties of matrices of zeros and ones. *Canadian J. Math. 9*, 371–377.

SANDERSON, J. 2000. Testing ecological patterns. *Amer. Sci. 88*, 332–339.

SNIJDERS, F. 1991. Enumeration and simulation methods for 0–1 matrices with given marginals. *Psychometrika 56*, 397–417.

TAN, P.-N., KUMAR, V., AND SRIVASTAVA, J. 2002. Selecting the right interestingness measure for association patterns. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 32–41.

TOMKINS, A. 2006. Private communication.

WANG, B. Y. AND ZHANG, F. 1998. Precise number of $(0, 1)$-matrices in $u(r, s)$. *Discrete Math. 187*, 211–220.

WEBB, G. 2006. Discovering significant rules. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 434–443.

WEBB, G. 2007. Discovering significant patterns. *Mach. Learn.*, to appear.

XIONG, H., SHEKHAR, S., TAN, P.-N., AND KUMAR, V. 2004. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, 334–343.

Received January 2007; revised June 2007; accepted July 2007