Geometric and Combinatorial Tiles in 0–1 Data

Aristides Gionis, Heikki Mannila, and Jouni K. Seppänen

Helsinki Institute for Information Technology,
University of Helsinki and Helsinki University of Technology, Finland

Abstract. In this paper we introduce a simple probabilistic model, hierarchical tiles, for 0–1 data. A basic tile (X, Y, p) specifies a subset X of the rows and a subset Y of the columns of the data, i.e., a rectangle, and gives a probability p for the occurrence of 1s in the cells of $X \times Y$. A hierarchical tile has additionally a set of exception tiles that specify the probabilities for subrectangles of the original rectangle. If the rows and columns are ordered and X and Y consist of consecutive elements in those orderings, then the tile is geometric; otherwise it is combinatorial. We give a simple randomized algorithm for finding good geometric tiles. Our main result shows that using spectral ordering techniques one can find good orderings that turn combinatorial tiles into geometric tiles. We give empirical results on the performance of the methods.

1 Introduction

The analysis of large 0–1 data sets is an important area in data mining. Several techniques have been developed for analysing and understanding binary data; association rules [3] and clustering [15] are among the most well-studied. Typical problems in association rules is that the correlation between items is defined with respect to arbitrarily chosen thresholds, and that the large size of the output makes the results difficult to interpret. On the other hand, clustering algorithms define distances between points with respect to all data dimensions, making it possible to ignore correlations among subsets of dimensions – an issue that has been addressed with subspace-clustering approaches [1, 2, 7, 8, 12].

One of the crucial issues in data analysis is finding good and understandable models for the data. In the analysis of 0–1 data sets, one of the key questions can be formulated simply as "Where are the ones?". That is, one would like to have a simple, understandable, and reasonably accurate description of where the ones (or zeros) in the data occur.

We introduce a simple probabilistic model, hierarchical tiles, for 0–1 data. Informally, the model is as follows. A *basic tile* $\tau = (X, Y, p)$ specifies a subset X of the rows and a subset Y of the columns of the data, i.e., a rectangle, and gives a probability p for the occurrence of 1 in the cells of $X \times Y$. A *hierarchical tile* τ consists of a basic tile plus a set of exception tiles, i.e., $\tau = (\tau_0, \{\tau_1, \ldots, \tau_k\})$, where each τ_i is a tile. The tiles $\tau_1, \ldots, \tau_k$ are assumed to be defined on disjoint subrectangles of τ_0. For an illustrative example, actually computed by our algorithm on one of our real data sets, see Figure 1. Given a point $(x, y) \in X \times Y$, the tile τ predicts the probability associated with τ_0, unless (x, y) belongs to the subset defined by an exception tile τ_i, for some $i \geq 1$; in this case the prediction is the prediction given by that particular τ_i. Thus a

J.-F. Boulicaut et al. (Eds.): PKDD 2004, LNAI 3202, pp. 173–184, 2004.

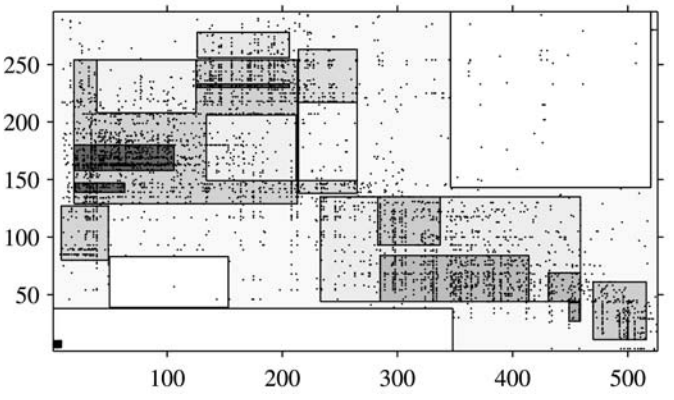

Fig. 1. Hierarchical tiling obtained for one of the data sets, `Paleo2`. The darkness of each rectangle depicts the associated probability.

hierarchical tile for which τ_0 covers the whole set $X \times Y$ defines a probability model for the set[1].

There are two types of tiles. If the rows and columns are ordered and X and Y are ranges on those orderings, then the tile is *geometric*; if X and Y are arbitrary subsets then the tile is *combinatorial*. Given a data set with n rows and m columns, there are $\Theta(n^2m^2)$ possible geometric basic tiles, but $\Theta(2^n2^m)$ possible combinatorial basic tiles. Thus combinatorial tiles are a much stronger concept, and finding the best combinatorial tiles is much harder than finding the best geometric tiles.

In this paper we first give a simple randomized algorithm for finding geometric tiles. We show that the algorithm finds with high probability the tiles in the data. We then move to the question of finding combinatorial tiles. Our main tool is spectral ordering, based on eigenvector techniques [9]. We prove that using spectral ordering methods one can find orderings on which good combinatorial tiles become geometric. We evaluate the algorithms on real data, and indicate how the tiling model gives accurate and interpretable results. The rest of the paper is organized as follows. In Section 2 we define formally the problem of hierarchical tiling, and in Section 3 we describe our algorithms. We present our experiments in Section 4, and in Section 5 we discuss the related work. Finally, Section 6 is a short conclusion.

2 Problem Description

The input to the problem consists of a 0–1 data matrix A with m rows R and n columns C. For row i and column j, the (i, j) entry of A is denoted by $A(i, j)$.

Rectangles. As we already mentioned, we distinguish between combinatorial and geometric rectangles. A *combinatorial rectangle* $r_c(A, X, Y)$ of the matrix A, defined for

[1] Our model can easily be extended to the case where each basic tile has a probability parameter for each column in Y; this leads the model to the direction of subspace clustering. For simplicity of exposition we use the formulation of one parameter per basic tile.

a subset of rows $X \subseteq R$ and a subset of columns $Y \subseteq C$, is a submatrix of A on X and Y. Geometric rectangles are defined assuming that the rows R and columns C of A are *ordered*. We denote such ordering by $R = \langle r_1, \ldots, r_m \rangle$ with $r_1 < \ldots < r_m$, where '$<$' is an ordering relationship. Given an ordering on R, a *range* X of R is a subset of *consecutive* rows of R. A *geometric rectangle* $r_g(A, X, Y)$ is now defined as the submatrix of A over the rows X and columns Y, where X and Y are ranges of R and C, respectively.

Tiles. To make the definition of hierarchical tiles noncircular we use a concept of the level, which tells how deep the nesting is. Given the data matrix A, a *basic tile*, or *level-0 tile* τ^0 is a rectangle r of A with an associated probability p, i.e., $\tau^0 = (r, p)$. Entries of A inside the rectangle r take value 1 with probability p and value 0 with probability $1 - p$. A *level-k tile* τ^k consists of a basic tile and a set of exception tiles; the exception tiles are of level at most $k-1$. We write $\tau^k = (\tau^0, \{\tau_1, \ldots, \tau_m\})$, where $\tau^0 = (r, p)$ and each τ_i is a tile of level at most $k - 1$. We require that the exception tiles $\tau_1, \ldots, \tau_m$ are *disjoint* and they are *contained* in τ^0. Finally, with each tile we associate a *domain*. The domain $\mathrm{Dom}(\tau^0)$ of a basic tile $\tau^0 = (r, p)$ is the rectangle r. The domain $\mathrm{Dom}(\tau^k)$ of a level-k tile $\tau^k = (\tau^0, \{\tau_1, \ldots, \tau_m\})$ is the domain of τ^0.

Prediction and likelihood. Given a position (i, j) in the data matrix A, the *prediction* $q(\tau^k, i, j)$ of a tile τ^k for (i, j) is defined recursively. For a basic tile $\tau^0 = (r, p)$ the prediction $q(\tau^0, i, j)$ is p (a basic tile predicts what it says). If $\tau^k = ((r, p), \{\tau_1, \ldots, \tau_m\})$, then $q(\tau^k, i, j) = p$, if $(i, j) \notin \bigcup_{l=1}^{m} \mathrm{Dom}(\tau_l)$ (if (i, j) is outside all exception tiles of τ^k). Otherwise, let t be the index such that $(i, j) \in \mathrm{Dom}(\tau_t)$; then $q(\tau^k, i, j) = q(\tau_t, i, j)$ (the prediction of the tile is the prediction of its exception that contains (i, j)). Let $A(r) = \{A(i, j) \mid (i, j) \in r\}$ be the restriction of data matrix A on the rectangle r. Given a tile $\tau^k = ((r, p), \{\tau_1, \ldots, \tau_m\})$ the likelihood of data $A(r)$ given τ^k is defined in the normal way:

$$L(A(r) \mid \tau^k) = \prod_{i,j} q(\tau^k, i, j)^{A(i,j)} (1 - q(\tau^k, i, j))^{1 - A(i,j)}.$$

Hierarchical tiling problem. The problem of finding hierarchical tiles that explain the data matrix as well as possible can now be formulated as finding the tile $\tau = ((A, p), \{\tau_1, \ldots, \tau_m\})$ that maximizes the likelihood $L(A \mid \tau)$. However, in order to avoid overfitting the data (very complex tiles that fit the data perfectly, e.g., using tiles at the level of single matrix entries) one needs to penalize for solutions with high complexity. Using Minimum Description Length (MDL) arguments, we define the *score* of $A(r)$ with respect to τ as $s(A(r) \mid \tau) = cK - \log L(A(r) \mid \tau)$, where K is a measure of total complexity of τ, and c is a scaling constant between complexity and minus log-likelihood. The complexity measure K is a function of the total number of tiles in τ – counting τ itself, its exceptions, the exceptions of its exceptions, and so on. If we denote the total number of tiles of τ by $|\tau|$ then K is defined to be $K = |\tau| \log |A(r)|$. The factor $\log |A(r)|$ is due to the fact that as the size of the data grows we need more information bits to specify the tiles, accounting to more complex models. The problem of finding hierarchical tiles can now be defined as follows: Given data matrix A, find the tile τ with the lowest score $s(A \mid \tau)$. The tile τ can be of any level, but it is required that $\mathrm{Dom}(\tau) = A$, i.e., it should cover the whole data matrix.

3 Algorithms

3.1 Geometric Tiles

We start describing our algorithm for discovering geometric tiles by first considering very simple cases, and then we discuss how to extend the ideas for the more complex situations. The simplest case is when we consider finding only one tile. Given a specific geometric rectangle $r = (A, \langle a, b\rangle, \langle c, d\rangle)$ of A to be used as the domain of the tile, the only choice to be made is the tile probability. As one can see easily, the maximum likelihood estimate for the tile probability is the frequency of the ones $f(r)$ in r. The frequency $f(r)$ can be computed in constant time, assuming that accumulating sums have been computed for all entries of the matrix: if $Ac(i, j)$ denotes the sum of 1s inside the rectangle $(A, \langle 1, 1\rangle, \langle i, j\rangle)$ then

$$f(r) = \frac{Ac(b, d) - Ac(b, c-1) - Ac(a-1, d) + Ac(a-1, c-1)}{(b-a+1)(d-c+1)}.$$

When the domain of the tile is not given, in principle one can try all possible rectangles r, evaluate the likelihood $L(A(r) \mid \tau(f(r), r))$ for each r, and select the tile that maximizes the likelihood. However, considering all rectangles is prohibitively expensive, since there are $\Theta(m^2n^2)$ different choices.

Designing an efficient algorithm to find a tile whose likelihood is provably not much worse than the likelihood of the best tile is a very interesting problem. However, it appears quite challenging: the likelihood function is not monotone with respect to tile containment, so there are no obvious ways to prune away potential tiles. Here we suggest a local-search algorithm for finding good tiles. The idea is to start with a random rectangle, and try to expand it or shrink it in each of the four directions. Expanding a rectangle r_0 in one direction, say to the right, is done in a sequence of geometric steps: for $r_0 = (\langle a, b\rangle, \langle c, d\rangle)$, we try all rectangles $(\langle a, b\rangle, \langle c, d+1\rangle)$, $(\langle a, b\rangle, \langle c, d+2\rangle)$, $(\langle a, b\rangle, \langle c, d+4\rangle)$, and so on, until the right boundary of the matrix is reached. The same expansion technique performed for other directions, and shrinking is done in a similar way. Out of all rectangles tried, the one with the largest likelihood is selected, call it r_1. If the likelihood of r_1 is larger than the likelihood of r_0, then a new expansion/shrinking phase starts from r_1. The process continues until a rectangle is found whose likelihood does not increase in an expansion/shrinking phase. A total of T random trials with different starting rectangles r_0 is performed, and the rectangle with the largest likelihood over all trials is given as the result.

Lemma 1. *Assume that the data matrix A contains i.i.d. bits with probability q, with the exception of one geometric rectangle R, which contains i.i.d. bits with probability $p \neq q$ and whose number of rows and columns is a constant fraction of the number of rows and columns (resp.) of the matrix A. Then, the local search method with random restarts will find R with high probability, i.e., probability bounded away from zero in the limit of infinite data.*

Due to space limitations the proof of all our claims is deferred to the full version of the paper.

Next we discuss how to find a larger collection of tiles with large likelihood. Our method employs the algorithm for finding one tile in a greedy fashion: Find the $(k+1)$-st tile with the best likelihood, given the k tiles that have been found so far. When searching for the next best tile, tiles that overlap existing tiles are not considered. This is checked during the expansion phase. To decide the number of tiles to be selected, the MDL score function $s(A \mid \tau)$ is used. When $s(A \mid \tau)$ stops decreasing, no more tiles are selected. For constructing the tile hierarchies, we have implemented and experimented with four different strategies:

Top down: At each step, the next tile is selected to be only at the same level, or included in already existing tiles.

Bottom up: The next tile is selected to be at the same level, or to include already existing tiles.

Mixed: The next tile is allowed to be anywhere as long as it does not overlap with existing tiles.

Single level: Only tiles of level 0 are selected.

Notice that the search space of the mixed strategy is the union of the search spaces of the other strategies, thus, one would expect the mixed strategy to outperform the others. The single-level strategy finds non-hierarchical tilings.

3.2 Combinatorial Tiles

In many applications, the rows and columns of the data set are not ordered, so it is important to be able to find combinatorial tiles. In this section we discuss our approach for this task. The basic idea is to transform the problem of finding combinatorial tiles to the previous case of finding geometric tiles. The transformation is done by ordering the rows and the columns of the data set, so that the rows and columns that are involved in combinatorial tiles become consecutive in the ordering. In this way, it is sufficient to search for geometric tiles in the ordered data set.

As we will see, it is not always possible to find such an ordering, since a data set might contain too many combinatorial tiles, and no single ordering can simultaneously transform all of them into geometric tiles. However, we will show that if a good ordering exists, our method will find it. The ordering method is based on the *spectral* properties of the data set. We next give a brief overview of the spectral techniques [9].

Consider a set of objects W and a symmetric matrix $S = (s_{ij})$ that specifies the similarity s_{ij} for each pair of objects (i, j). The *Laplacian matrix* of S is defined as the symmetric and zero-sum matrix $L_S = D_S - S$, where D_S is the diagonal matrix whose (i, i)-th entry is the sum of the i-th row (or column) of S, that is, $d_i = \sum_j s_{ij}$. Let e be the vector having value 1 in all its entries. Since all rows (and columns) of L_S sum to zero, we have $L_S e = 0$, which means that e is an eigenvector of L_S, corresponding to the eigenvalue 0. Because L_S is a symmetric positive semidefinite matrix, all of its eigenvalues are real and nonnegative, and therefore 0 is the smallest eigenvalue. The second smallest eigenvalue of L_S is called the *Fiedler value*, and the corresponding vector is called *Fiedler vector* [11]. One can show that the Fiedler value is given by

$$\min_{\substack{x^T e=0 \\ x^T x=1}} x^T L_S x = \min_{\substack{x^T e=0 \\ x^T x=1}} \sum_{i,j} s_{ij}(x_i - x_j)^2 \tag{1}$$

and the Fiedler vector is a vector that achieves the minimum subject to the constraints $x^T e = 0$ and $x^T x = 1$. A vector x can be viewed as *mapping* from objects in W to real numbers. In particular, the object i in W is mapped to the i-th coordinate x_i of x. If we view Equation (1) as an energy function $F_S(x) = x^T L_S x$, then the Fiedler vector v has the property that it minimizes $F_S(x)$ over all vectors x that satisfy the constraints $x^T e = 0$ and $x^T x = 1$. Intuitively, because of the terms $s_{ij}(x_i - x_j)^2$, minimizing $F_S(x)$ results to mapping "similar" objects to "near-by" values. The two constraints have a simple interpretation: $x^T e = 0$ requires the vector x to be orthogonal to the trivial solution vector e, i.e., to have zero mean, and $x^T x = 1$ amounts to fixing the scale of the solution, i.e., the variance of x is 1.

Our method uses Fiedler vectors to order the rows and the columns of a data set A. The idea is to consider each row as an "object" and define the similarity matrix $S = (s_{ij})$ for pairs of rows. Two natural definitions of row similarity is the *Hamming similarity* and *dot-product similarity*. The Hamming similarity h_{ij} between rows i and j is the number of common values, while the dot-product similarity c_{ij} is defined to be the number of common 1s. Both similarity definitions are used in our experiments. The method computes the row-row similarity matrix S for the data matrix A, and then it computes the Fiedler vector of the Laplacian L_S. The rows are ordered on the basis of their Fiedler-vector coordinates. The columns are ordered with the same method, independently of the rows.

Next we will show that for a data set generated from a simple combinatorial tiling model, the spectral algorithm will discover the correct structure. We begin with some definitions. An $n \times n$ matrix S has (k, d, s_1, s_2)-*block structure* if the n indices of S can be partitioned in k blocks $B_1, \ldots, B_k$ as follows: (i) the size of each block is greater than d, (ii) for all $i, j \in B_l$ we have $s_{ij} = \tilde{s}_l \geq s_1$, i.e., the value s_{ij} for indices within a block is a constant greater than s_1, (iii) for all $i \in B_l$ and $j \in B_t$ with $l \neq t$ we have $s_{ij} = \tilde{s}_{lt} \leq s_2$, i.e., the value s_{ij} for indices in different blocks is a constant smaller than s_2. We say that a vector x *respects* the structure of a (k, d, s_1, s_2)-block matrix S if for every triple of indices (i, j, h) with $x_i \leq x_j \leq x_h$ it cannot be the case that $i \in B_l$, $j \in B_t$, $h \in B_g$, and $l = g \neq t$.

Lemma 2. *Let S be an $n \times n$ object-similarity matrix that has $(2, d, s_1, s_2)$-block structure. Consider $S' = S + E$, where E is a symmetric matrix. Then the Fiedler vector of $L_{S'}$ respects the structure of S, provided that $|L_E| = o(d(s_1 - s_2))$, where $|L_E|$ is the norm of the matrix L_E.*

The situation is more complex when the similarity matrix has more than 2 blocks, as the associated Fiedler vectors form a subspace of dimension greater than 1. For example, consider a matrix with 3 blocks. A Fiedler solution is to assign all objects at three distinct values: one value for all objects within the same block. A different Fiedler solution uses only two distinct values: one value for objects in the first and second blocks and one value for objects in the third block. Note that Lemma 2 does not hold for the second solution, because if we break ties arbitrarily, most likely the objects of the first two blocks will not respect the block structure of the matrix.

To address the problem that in a Fiedler solution more than one blocks might be mapped to the same value, we have used a *recursive* application of the spectral algorithm: we first divide the coordinates of the Fiedler vector in two groups so that the sum

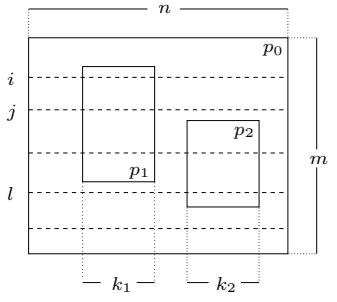

Fig. 2. Illustrative example of a data set generated from a "simple" tiling model.

1 0 0 1
1 0 1 0
0 1 1 0
0 1 0 1

Fig. 3. A case in which no ordering can turn all combinatorial rectangles into geometric rectangles.

of variances of the two groups is minimized, and then we apply recursively the spectral method in each of the two groups. A recursive step is performed only if the sum of variances in the two groups is less than half of the variance of all coordinates. Since the two groups that minimize the sum of variances cannot overlap, the groups can be determined optimally by sorting the coordinates and searching for the best breakpoint. We believe that using the recursive application of the spectral algorithm, Lemma 2 can be proven for matrices with more than two blocks, and this was verified in our experiments.

We complete our argument by showing that for a data set generated from a "simple" tiling model, the conditions of Lemma 2 hold. Consider, for example, the case of the simple data set shown in Figure 2: a 1 is generated at each entry of the data set with probability p_0, except in the two tiles, where a 1 is generated with probabilities p_1 and p_2, respectively. The tiles considered are combinatorial and the task is to reorder the rows and columns so that the geometric tiles shown in the figure emerge.

Consider the row-row similarity matrix S, where the Hamming similarity is used; a similar argument can be made for the dot-product similarity. A block is defined by the set of rows that intersect the same tiles, for example, rows i and j in Figure 2 belong to the same block. If the probability of the data at the entry A_{ih} is p_{ih}, then the similarity s_{ij} between rows i and j can be written as a sum of independent Bernoulli trials, i.e., $s_{ij} = \sum_{h=1}^{n} W_{ijh}$, where W_{ijh} is 1 with probability $p_{ijh} = p_{ih}p_{jh} + (1 - p_{ih})(1 - p_{jh})$ and 0 otherwise. Then, the *expected* similarity between rows i and j is $E[s_{ij}] = \sum_{h=1}^{n} p_{ijh}$. For example, the expected similarity between rows i and j in Figure 2 is $E[s_{ij}] = (n - k_1)(p_0^2 + (1 - p_0)^2) + k_1(p_1^2 + (1 - p_1)^2)$. We define the "simple" tiling model by making the following assumptions.

(i) Each block consists of a constant fraction of the total number of rows, i.e., $\Theta(m)$.

(ii) For each row i, the expected similarity $E[s_{ij}]$ is maximized for rows j in the same block. This assumption is reasonable in many case, for example, in Figure 2 it holds if, say, p_0 is less than 1/2, and p_1 and p_2 are greater than p_0. We assume that the expected similarity of two rows i and j in the same block is $E[s_{ij}] \geq s_1$, while the expected similarity of two rows i and j in different blocks is $E[s_{ij}] \leq s_2$. Furthermore, we assume that $s_2 - s_2 = \Theta(n)$. Again this assumption holds for the situation depicted in Figure 2 for, say, $p_0 = 0.2$, $p_1 = 0.8$, $p_2 = 0.7$.

Theorem 1. *For a simple tiling model as described above, the spectral algorithm will discover the correct ordering or rows and columns with high probability. The probability is taken over the generation of particular data instances from the model.*

In more complicated situations, it is possible that there is no ordering that turns all combinatorial tiles in the data into geometric tiles simultaneously. For example, the matrix in Figure 3 has four combinatorial tiles of 1s of size 2×1, but no reordering can bring these tiles together as geometric tiles. A possible solution to this problem is to first find the best tile in one ordering of the data, then reorder in a way that disregards the tiles already found, and then continue finding tiles. We feel that this approach would seriously detract from the interpretability of the results, so we restrict ourselves to finding tiles in one ordering only.

4 Experimental Evaluation

We used four real data sets to test our tiling algorithms. The first two sets contain information about fossil findings: the rows correspond to sites and the columns to genera. The first, `Paleo1`, contains 124 sites and 139 genera, and the second, `Paleo2`, 526 sites and 296 genera. The third data set, `Course`, contains information about Masters-level course registrations at the University of Helsinki Department of Computer Science. The set has 102 courses and 1739 students, with an average of 3.0 course registrations per student. For the fourth data set, `Movie`, we took the smaller of the two MovieLens movie-rating data sets[2], and turned it into a 0–1 matrix by mapping the high ratings 4 and 5 to 1, and the lower ratings (and non-ratings) to 0. The resulting data set contains ratings on 1682 movies by 943 users, with an average of 58.7 movies per user.

For each of the data sets, we first reordered both the rows and the columns by spectral ordering as outlined in Section 3.2, using both cosine and Hamming similarity as the similarity function. Then we ran the tiling algorithm described in Section 3 until it had found 50 tiles, using 100 random restarts per tile. The algorithm has four alternatives for the search strategy: top-down, bottom-up, mixed, and single-level.

Figure 4 shows how the log-likelihood behaves as a function of the number of tiles. Plots are shown for all the data sets and all strategies applied, but only for the cosine similarity. We see that mixed and top-down strategies outperform bottom-up and single-level. The reason that top-down is better than bottom-up and as good as mixed is probably the following: since we start selecting tiles greedily so that the total likelihood becomes as large as possible, we favor large tiles in the beginning, and it is thus more beneficial to recurse into those tiles than combining them to form even larger tiles.

Figure 5 shows some examples of tilings found with the different strategies. The top-down strategy found tilings very similar to those of the mixed strategy, and single-level was very similar to bottom-up. The probabilities of the tiles are shown in shades of grey, so that white corresponds to 0 and black to 1. The figure supports our hypothesis of why top-down and mixed outperform bottom-up and single-level: all strategies have found some large almost-empty tiles, but mixed and top-down can recurse into them to find the exceptions, whereas bottom-up and single-level only keep tiling the untiled

[2] http://www.grouplens.org

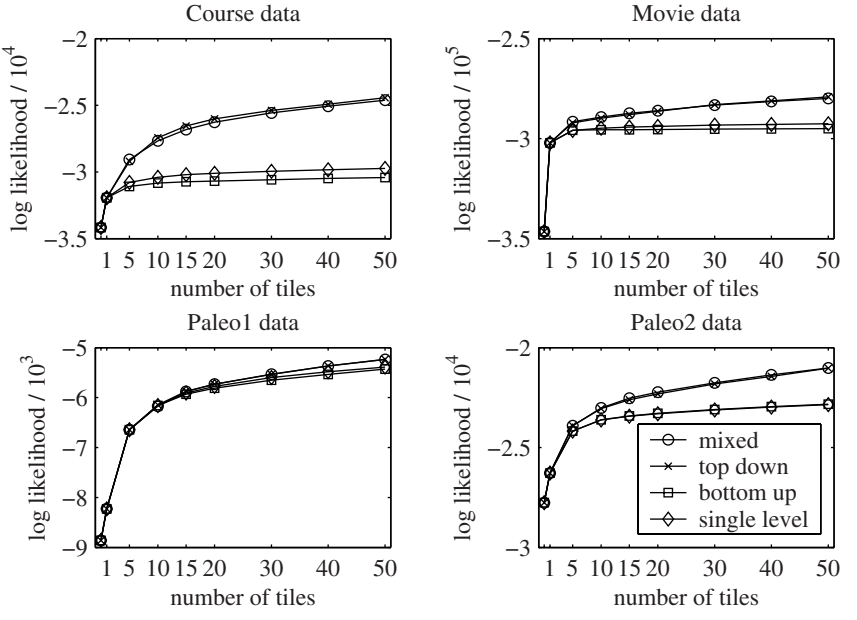

Fig. 4. Log-likelihood of model as a function of the number of tiles.

area, which has fewer opportunities. The tilings found on the Hamming-sorted data look somewhat more balanced than those of the cosine-sorted data, since the Hamming measure has grouped dense subsets in two corners of the matrix.

One parameter in our algorithm is the number T of the number of random restarts used when selecting each tile. To assess the effect of this parameter, we varied its value and computed the total log-likelihood of 10-tile models for two data sets, `Paleo2` and `Course`. The results are shown in Figure 6; each log-likelihood value shown is the average from 200 runs. As is to be expected, there is a diminishing-returns phenomenon, and after some point it helps very little to increase the number of restarts.

As an example of the interpretability of the results, we show one tile and its smaller exception tile in the `Course` data. The larger tile has 308 students and the following 11 courses, and its probability is 2.9% (which is relatively high compared to the background tile's 0.1%):

- User interface research
- Object architectures
- Simulation methods
- Implementation of the Linux system
- Object databases (*)
- Architectures of object systems (*)
- Research course in object languages
- Computer-aided co-operation
- Mobile workstations
- Algorithm technology
- Object technology

The exception tile consists of 115 students and the two courses marked with (*) above, and has probability 17.0%. Thus, there is a tight core of students interested in object-oriented technologies, and around that core there are more students who study object-oriented methodology, user interfaces, and some applications. (The probabilities

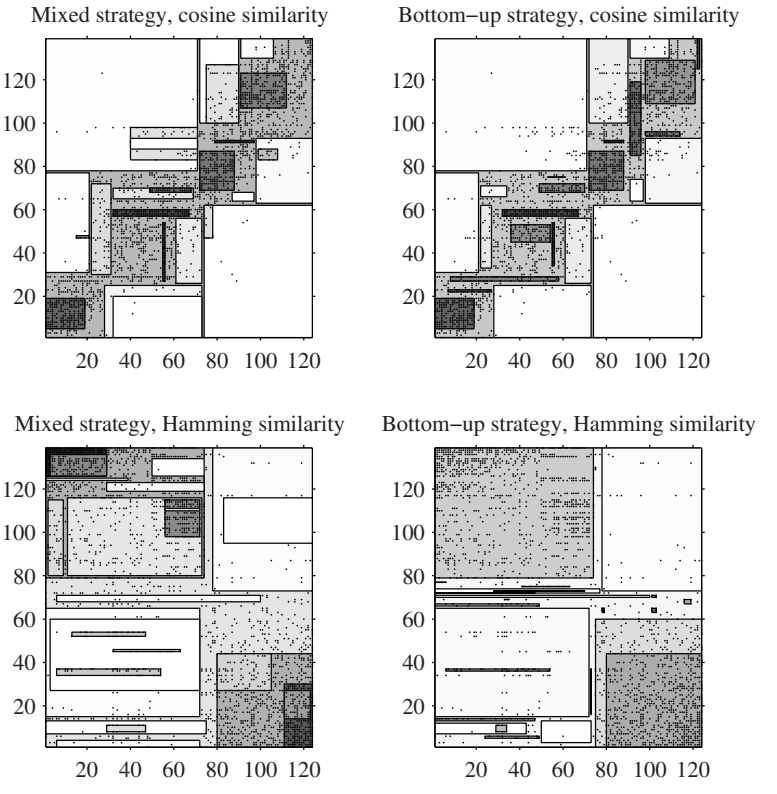

Fig. 5. 25-tile models of the `Paleo1` data found using two of the four different search strategies and two different orderings of the matrix.

may seem low, but many of these courses are in fact small seminars that have only been organized once.)

5 Related Work

Hierarchical tiles are partly motivated by the classical work of Rivest on finding decision lists [22]. Related is also the work on ripple-down rules [13]. A PAC-learning algorithm for finding hierarchical concepts was given in [16].

A lot of work on the analysis of 0–1 data has focused on finding frequent item sets and association rules [3, 6, 14]. A tile can be viewed as a frequent itemset: the tile's columns are the the items and the rows are the supporting transactions. A key difference with these approaches is that our method allows for errors, and also that a tile with many items might be selected even if its support is low. In addition, the greedy nature of our algorithm allows the user to look only at the first k tiles found, and interpret them as the tiles that best explain the data set among all tilings of size k.

A related problem is that of finding maximal empty rectangles in data [10, 18]. The crucial difference to our task is that we do not require tiles to be completely empty

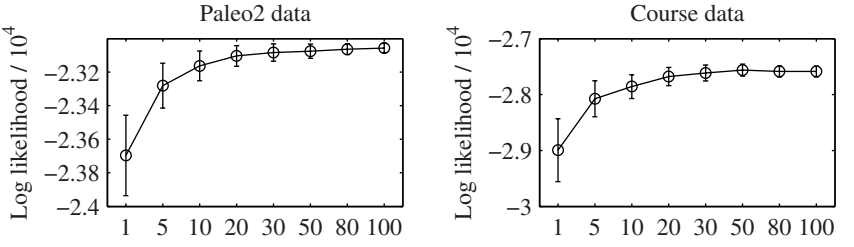

Fig. 6. Log-likelihood of 10-tile models for `Paleo2` and `Course` data as a function of the number of restarts. The circles denote the average of 200 runs, and the error bars indicate the standard deviation. The strategy was always "mixed", but different random numbers were used.

or completely full, although such tiles do have maximal likelihood among otherwise similar tiles.

As mentioned in the introduction, our results generalize immediately to the case where the tiles specify individual probabilities for each column. That is, define a generalized basic tile to be a triple $(X, Y, \bar{p})$, where X is a subset of the columns, Y is a subset of the rows, and $\bar{p}$ associates a probability p_A for each $A \in Y$. All other definitions are changed in a straightforward manner. This extension brings our model quite close to subspace clustering [1, 2, 7, 8, 12, 19].

Spectral algorithms are important tools for many application areas and they have been used in a wide range of problems, such as, solving linear systems [21], ordering problems [4, 17], data clustering [20, 23], and other. One way to perform the sorting more efficiently is to apply the spectral technique only to a subset of the data and then to refine the ordering of the whole data. [5]

6 Concluding Remarks

We have defined the concept of hierarchical tiles, and shown how they give a natural probabilistic model for 0–1 data. We gave a simple algorithm for finding geometric tiles, and showed some of its properties. We discussed the use of spectral ordering methods for finding good orderings. Our main theoretical result is that under certain assumptions the orderings produced by spectral techniques are such that strong combinatorial tiles become actually geometric tiles in the ordering. We demonstrated the applicability of the notion of hierarchical tiles by giving example results on real data.

References

1. C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, 2000.
2. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.
3. R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in large databases. In *SIGMOD*, 1993.

4. J. Atkins, E. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1), 1999.
5. A. Beygelzimer, C.-S. Perng, and S. Ma. Fast ordering of large categorical datasets for better visualization. In *SIGKDD*, 2001.
6. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *PKDD*, 2002.
7. C. Cheng, A. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, 1999.
8. Y. Cheng and G. Church. Biclustering of expression data. In *ISMB*, 2000.
9. F. Chung. *Spectral graph theory*. American Mathematical Society, 1997.
10. J. Edmonds, J. Gryz, D. Liang, and R. J. Miller. Mining for empty spaces in large data sets. *Theor. Comput. Sci.*, 296(3):435–452, 2003.
11. M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23, 1973.
12. J. Friedman and J. Meulman. Clustering objects on subsets of attributes. *JRSS B*, 2004.
13. B. Gaines and P. Compton. Induction of ripple-down rules applied to modeling large databases. *JIIS*, 5(3), 1993.
14. J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, 2002.
15. A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 1999.
16. J. Kivinen, H. Mannila, and E. Ukkonen. Learning hierarchical rule sets. In *COLT*, 1992.
17. Y. Koren and D. Harel. Multi-scale algorithm for the linear arrangement problem. Technical Report MCS02-04, The Weizmann Institute of Science, 2002.
18. B. Liu, L.-P. Ku, and W. Hsu. Discovering interesting holes in data. In *IJCAI*, 1997.
19. T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pac. Symp. Biocomp.*, volume 8, 2003.
20. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.
21. A. Pothen, H. Simon, and L. Wang. Spectral nested dissection. Technical Report CS-92-01, Pennsylvania State University, Department of Computer Science, 1992.
22. R. Rivest. Learning decision lists. *Machine Learning*, 2(3), 1987.
23. H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. In *CIKM*, 2001.