Periodica Mathematica Hungarica Vol. 50 (1-2), 2005, pp. 199–221

CLUSTERING METHODS IN MICROARRAYS

Lídia Rejtő (Budapest, Newark) and
Gábor Tusnády (Budapest)

Dedicated to Endre Csáki and Pál Révész on the occasion of their 70^{th} birthdays

Abstract

DCT Given a finite set of points in an Euclidean space the *spanning tree* is a tree of minimal length having the given points as vertices. The length of the tree is the sum of the distances of all connected point pairs of the tree. The *clustering tree with a given length* of a given finite set of points is the spanning tree of an appropriately chosen other set of points approximating the given set of points with minimal sum of square distances among all spanning trees with the given length.

DCM A matrix of real numbers is said to be *column monotone orderable* if there exists an ordering of columns of the matrix such that all rows of the matrix become monotone after ordering. The *monotone sum of squares of a matrix* is the minimum of sum of squares of differences of the elements of the matrix and a column monotone orderable matrix where the minimum is taken on the set of all column monotone orderable matrices. *Decomposition clusters of monotone orderings* of a matrix is a clustering of the rows of the matrix into given number of clusters such that the sum of monotone sum of squares of the matrices formed by the rows of the same cluster is minimal.

DCP A matrix of real numbers is said to be *column partitionable* if there exists a partition of the columns such that the elements belonging to the same subset of the partition are equal in each row. Given a partition of the columns of a matrix the *partition sum of squares of the matrix* is the minimum of the sum of square of differences of the elements of the matrix and a column partitionable matrix where the minimum is taken on the set of all column partitionable matrices. *Decomposition of the rows of a matrix into clusters of partitions* is the minimization of the corresponding partition sum of squares given the number of clusters and the sizes of the subsets of the partitions.

Mathematics subject classification number: 05C05, 05C15, 62H30.

Key words and phrases: clusters, microarray, tree, regularity lemma, partitions, network.

Talk presented on June 17, 2004, at the conference in celebration of the 70^{th} birthday of Endre Csáki and Pál Révész

0031-5303/2005/$20.00
Akadémiai Kiadó, Budapest
© Akadémiai Kiadó, Budapest
Springer, Dordrecht

Contents

1. Introduction

The typical data structure in statistical investigation is a matrix. Data that we have in mind have features in common with a questionnaire. The rows are the questions that the investigator poses concerning an investigated phenomena, and the columns correspond to the answers of the different subjects. In microarray analysis the questions are genes or clones, and the answers are the expression levels of the genes in different cell types or under different conditions ([5], [6], [20], [25], [26]). Clustering of microarray measurements means grouping the elements by rows and columns. Grouping by rows reveals the structure and organization of the basic elements of the cell (the enzymes, the membranes, the energy buffers, etc.), while grouping by columns leads to an understanding of the dynamics of life in the cell.

Working with microarray measurements we developed new clustering methods that offer novel insights into the structure of genes and their dynamics. In this paper we present three different methods: tree clustering, monotone clustering, and partition clustering. Each has its special power to uncover some hidden property of the data. Tree clustering is a special case of the deconvolution problem where an unknown multidimensional distribution is estimated from a sample corrupted by additive Gaussian noise. Partition clustering is an extension of a stochastic model used in graph theory and it is related to simultaneous clustering of genes and conditions developed in recent years. The idea of monotone clustering is new, thus we describe it in more details.

2. Decomposition of clusters forming a tree

2.1. Deconvolution problem

Let T be an *Euclidean tree* in the p-dimensional Euclidean space, it is a union of connected multidimensional segments without cycles:

$$T = \bigcup_{(a,b)\in\mathcal{E}} S(V(a), V(b)),$$

where $\mathcal{E}$ is a set of edges and $\mathcal{V}$ is a finite set of vertices. The abstract set $\mathcal{V}$ is represented by the set of multidimensional points V and the segment $S(V_0, V_1)$ is the multidimensional interval

$$S(V_0, V_1) = \bigcup_{0\le t\le 1} \{V_t = (1-t)V_0 + tV_1\}$$

joining the points V_0, V_1. We call the Euclidean tree T the *representation* of the abstract or graph theoretical tree consisting of $\mathcal{V}$ and $\mathcal{E}$. Let X be the uniform distribution on T, that is X is a mixture of uniform distributions on the segments of T with mixing probabilities proportional to the length of the segments. Let us suppose that we have an $X_1, \dots, X_n$ iid sample from X but the sample elements cannot be observed directly, instead we have a sample

$$Y_i = X_i + \sigma Z_i, \qquad i = 1, \dots, n,$$

where σ is a positive constant and the Z_i-s are independent standard normal random variables, that is the coordinates of the Z_i-s are independent one dimensional Gaussian random variables, and $(X_1, \dots, X_n)$ and $(Z_1, \dots, Z_n)$ are independent.

The *deconvolution problem* is to estimate T from a sample $Y_1, \dots, Y_n$. This is a special case of a general problem where the distribution of X is not specified. In the general setting consider the set of n-dimensional vectors $\mathcal{F} = (f(Y_1), \dots, f(Y_n))$, where f is the density function of the sample elements. The set $\mathcal{F}$ is convex and bounded in the n-dimensional Euclidean space. The convexity follows from the fact that we can swap the mixing of vectors for mixing of X-distributions.

The log-likelihood function $\mathcal{L} = \sum_{i=1}^{n} \log(f(Y_i))$ is a strictly concave function on $\mathcal{F}$. Thus the n-dimensional vector f^* corresponding to the maximum likelihood estimator is uniquely determined. According to Minkowski theorem, as any point of $\mathcal{F}$, f^* is a convex combination of $n+1$ extremal points of $\mathcal{F}$. The extremal points of $\mathcal{F}$ correspond to X-s concentrated on one single point. Thus the maximum likelihood estimator of the unknown distribution of X is concentrated on $n+1$ points.

To find numerically the maximum likelihood estimator the EM-algorithm applies. For large sample size n we can use the uniform distribution on the sample, that is the empirical distribution of the sample, as a starting distribution. Perhaps, the empirical characteristic function can be used, however, further properties are unknown. We met this problem in astronomy ([1]), where the unknown distribution of X is located on an unknown segment. The speed of convergence of the estimator of the unknown distribution of X was determined by Halász [14]. Later his theorem was rediscovered in paper [28]. The speed of convergence is surprisingly slow, it is $1/\log n$ instead of $1/\sqrt{n}$. The case when X is uniformly distributed on a simplex is discussed by Perczel *et al.* [21], decomposition of mixtures problems of army locators was discussed in [22].

In the previous discussion the variance σ was supposed to be known which is not the case in practice. It can be estimated in the usual way as the sum of square of errors but it can not be too small: our proposition is $10\Delta/n$ as a lower bound for σ where Δ is the diameter of the sample. The overestimation of σ has a smoothing effect on the distribution of X. We have no version of the EM-algorithm applicable to direct estimation of the unknown tree, hence we substitute the maximum likelihood method with the least square one. We approximate the likelihood of a sample point by

$$\frac{1}{L\sigma^{p-1}} \exp\left(-\frac{d^2}{2\sigma^2}\right),$$

where L denotes the length of the tree, p is the dimension, and d is the distance of the sample point from the tree.

2.2. Least square estimator

Imagine that the sample elements Y_i, $i = 1, \dots, n$ are towns in a multidimensional universe and the unknown tree is a highway system to be built. The towns will be connected to the highway system with segments $S(Y_i, U_i)$, where U_i is the closest point of the highway system to the town Y_i. The total *travelling cost* is

$$C(T) = h \log(L) + \frac{1}{n} \sum_{i=1}^{n} d^2(Y_i, U_i),$$

where $L = \sum_{(a,b)\in\mathcal{E}} d(V(a), V(b))$ is the total length of the highway, d denotes the Euclidean distance, and h is a positive constant. The *highway problem* is to minimize $C(T)$ for given Y_i-s and h.

Given a tree T a next aspirant is the spanning tree of the set $U_1, \ldots, U_n$ to minimize $C(T)$. Without any smoothing this step is not effective in itself. A possible smoothing is the use of trees with fixed number k of vertices, where k is much more smaller then n. Fixing the number of vertices any optimization method is applicable while we tacitly use the spanning tree of vertices. An initial set of vertices is provided by k-means clustering. In a certain sense tree clustering is an extension of k-means clustering: in both cases the sample elements are approximated by a structured center. In k-means clustering the center is a finite set and in tree clustering it is the tree. In McQueen's algorithm for k-means clustering two steps are alternating:

– *assignment:* having a center the sample elements are ordered into new clusters
– *centering:* having a clustering of the sample elements the new cluster centrums are calculated as averages.

In tree clustering the assignment step is finding the closest points U_i to the sample elements Y_i on the tree T. We do not see how can we build up a new tree with given number of vertices from the U_i-s. Each U_i is located on one edge (a_i, b_i) of T and it is defined by a parameter $0 \leq t_i \leq 1$ on $S(V(a_i), V(b_i))$. The centering problem is to find an appropriate representation of T using the abstract set $\{(a_i, b_i, t_i), i = 1, \ldots, n\}$.

2.3. Testing tree structures

Although maximum likelihood estimator and a minimum of the cost function $C(T)$ under given constraints exist for any set of points the hypothesis that the distribution of X is uniform on a tree has to be tested. The test statistics is the difference between the free and the tree maximum likelihoods. In practice we generate the cut point of the statistics applying bootstrapping.

2.4. Reconstruction of Kauffman networks applying trees

The tree structure as dendrogram appears in hierarchical clustering. Technically a dendrogram is different from tree clustering, but in spirit both methods lead to similar structure of the clusters. The ultimate goal in microarray analysis is to discover gene networks and the interaction structure of genes. In paper [24] we considered 280 different genes under 28 different conditions and the structure built by tree clustering was used for gene network building. In Kauffman's network theory ([16], [23]) gene interactions are modelled by Boolean functions. Interestingly the tree structure appears in the Boolean functions as well (see [17], where the new class of Boolean functions is called nested canalyzing ones).

3. Decomposition clusters of monotone orderings

3.1. Numbering the columns

A matrix is a real valued function on the Cartesian product of two finite sets. Let us denote the first set by $\mathcal{R}$, it is the set of rows, and the second one by $\mathcal{C}$, which is the set of columns. For any pair of element $r \in \mathcal{R}$, $c \in \mathcal{C}$ let us denote by $a(r, c)$ the value of the function, which is the element of the matrix A in row r and column c. Let us denote the number of elements of the set $\mathcal{R}$ by n and that of $\mathcal{C}$ by p. A *numbering of the columns* of a matrix is an invertible function mapping the set $\mathcal{C}$ into the set of the first p positive integers:

$$\pi : \mathcal{C} \to \{1, \dots, p\}.$$

We denote the number assigned to the element $c \in \mathcal{C}$ by $\pi(c)$. For all $c \in \mathcal{C}$, $\pi(c)$ is an integer between 1 and p and any integer between 1 and p appears exactly once among the numbers $\pi(c)$, the function π is one-to-one. Let us denote by c_i the element of $\mathcal{C}$ for which $\pi(c_i) = i$, $1 \le i \le p$. We say that a *numbering* π *is monotone* if the sequences

$$t_i = a(r, c_i), \quad i = 1, \dots, p$$

are monotone for all $r \in \mathcal{R}$. We say that a matrix is *column monotone orderable* if there exists a monotone numbering of the columns of the matrix.

Observe that our terminology incorporates some ambiguity coming from the fact that usually we imagine the columns and rows of a matrix as something being already ordered. When they are ordered, the monotone numbering means a reordering. The crucial step here is the numbering: in the beginning the columns are labelled by a simple character c, after the numbering we denote them by c_i. Accordingly, we imagine that the set of columns has no order at all. The ordering or numbering is our task. If there is a monotone numbering then, in a certain sense, it is a natural order of the columns.

Let us denote the set of column monotone orderable matrices by $\mathcal{M}(\mathcal{C}, \mathcal{R})$. The *monotone sum of squares* of a matrix A is the minimum of the sum

$$\sum_{r \in \mathcal{R},\, c \in \mathcal{C}} (a(r, c) - m(r, c))^2$$

taken on $M \in \mathcal{M}(\mathcal{R}, \mathcal{C})$, where $m(r, c)$ stands for the element of M in row r and column c. We denote this quantity by $W(A)$:

$$W(A) = \min_{M \in \mathcal{M}(\mathcal{R}, \mathcal{C})} \sum_{r \in \mathcal{R},\, c \in \mathcal{C}} (a(r, c) - m(r, c))^2.$$

Monotone sum of squares of column monotone orderable matrices is zero. For other matrices it is strictly positive. We shall denote by $\pi(A)$ any numbering with monotone sum of squares $W(A)$. This is not uniquely defined. We shall refer to $\pi(A)$ as *optimal numbering* of A. One natural ambiguity here comes from monotonicity

itself: a monotone sequence may be either decreasing or increasing. Thus for any monotone numbering π the numbering

$$\nu(c) = p + 1 - \pi(c), \quad c \in \mathcal{C}$$

is also monotone. If $\mathcal{C}$ itself is the set of the first p positive integers, then the numbering π is a permutation and has an inverse π^{-1} that is defined by $\pi^{-1}(\pi(i)) = i$ for all $1 \leq i \leq p$.

If the set of rows $\mathcal{R}$ has only one element, then the matrix is called a row vector. Any matrix with only one row is monotone orderable. For any numbering π of the columns of the matrix A and any row

$$t = (t_i = a(r, c_i),\ i = 1, \ldots, p)$$

listing the elements of the row according to the numbering π. The *monotone regression* of the sequence t is the monotone sequence $m = (m_i, 1 \leq i \leq p)$ minimizing the quadratic error

$$\sum_{i=1}^{p} (t_i - m_i)^2.$$

We shall denote the minimum by $w_\pi(r)$ and the sum of the minimums taken for all rows will be denoted by $W_\pi(A)$:

$$W_\pi(A) = \sum_{r \in \mathcal{C}} w_\pi(r) = \sum_{r \in \mathcal{C}} \min_m \sum_{i=1}^{p} (t_i - m_i)^2.$$

We gather the minimizing row vectors m into a matrix M_π. This matrix will be referred as the *monotone approximation of A with respect to the ordering π*. As it is easily seen, $W(A)$ is the minimum of $W_\pi(A)$ taken on all numberings. The monotone regressions of the rows according to the optimal numbering $\pi(A)$ form together the monotone matrix M minimizing the sum $\sum_{r \in \mathcal{R}, c \in \mathcal{C}} (a(r, c) - m(r, c))^2$.

Clustering of the rows of a matrix is a function ρ mapping the set $\mathcal{R}$ of the rows of the matrix into the set of first k positive integers:

$$\rho : \mathcal{R} \to \{1, \ldots, k\},$$

where k, the number of clusters is a given number between 1 and n. We say that k is the size of ρ. Let us denote the number assigned to the element $r \in \mathcal{R}$ by $\rho(r)$. All the numbers $\rho(r)$ are integers between 1 and k and all integers between 1 and k appear at least once among the numbers $\rho(r)$, the function ρ is not a one-to-one mapping of $\mathcal{R}$ to the set $\{1, \ldots, k\}$, if $k < n$. For any $1 \leq i \leq k$ let us denote by $\mathcal{R}_i$ the subset of $\mathcal{R}$ having the number $i : \rho(r) = i$ for all $r \in \mathcal{R}_i$. Let us denote by A_i the matrix formed by the rows $r \in \mathcal{R}_i$. As a function A_i of the Cartesian product of $\mathcal{R}_i$ and $\mathcal{C}$ is a restriction of the function A on the Cartesian product of $\mathcal{R}$ and $\mathcal{C}$: it has the same columns as A but it contains only the rows numbered by i.

Decomposition clusters of monotone orderings of the matrix A is a clustering of the rows of A minimizing the sum

$$C(\rho) = \sum_{i=1}^{k} W(A_i)$$

on the set of all clusterings with size k, where $W(A_i)$ is the monotone sum of squares of the submatrix A_i of A defined by the clustering.

The function $W(A_i)$ implicitly refers to the optimal numbering $\pi_i = \pi(A_i)$ of the columns of A_i. Observe that the optimal numberings π_i, $i = 1, \ldots, k$ belonging to different clusters need not be the same. Actually, in practical cases they are rather different. In clusterings the number of clusters is a quantity to be minimized, too. Let us denote by $\mathcal{W}_k(A)$ the minimum of $C(\rho)$ on clusterings with size k. As it is easily seen, this is a monotone decreasing sequence:

$$\mathcal{W}_1(A) \geq \mathcal{W}_2(A) \geq \ldots \geq \mathcal{W}_n(A) = 0.$$

There is a basic asymmetry in the whole setup here between rows and columns. Numbering of the columns is one-to-one and the order represented by the numbering is crucial. We can use any mapping of $\mathcal{C}$ onto the real numbers having the only property that different columns have different numbers. Clustering of the rows is actually a partition. The order of the clusters formed by the rows has no importance. We shall take any pair of clusterings resulting the same partition of rows to be equivalent. In this sense there exists only one clustering for $k = n$ resulting row vector submatrices A_i having zero monotone sum of squares.

3.2. Monotone regression

Monotone regression is minimization of

$$\sum_{i=1}^{p} (t_i - m_i)^2$$

for given positive number p and sequence $t = (t_1, \ldots, t_p)$ on monotone sequences $m = (m_1, \ldots, m_p)$. For monotone increasing m the algorithm is the following. The monotone regression itself is a clustering of the real numbers t_i into consecutive clusters having monotone increasing averages. Starting with

$$\mathrm{bll}_1 = 1, \quad \mathrm{mve}_1 = t_1$$

for all $i = 2, \ldots, p$ we take the following steps: we start with

$$\mathrm{bll}_i = 1, \quad \mathrm{mve}_i = t_i.$$

If $t_i < \mathrm{mve}_{i-1}/\mathrm{bll}_{i-1}$ then set $j = i - 1$ and repeat the following steps. If

$$\mathrm{mve}_i/\mathrm{bll}_i < \mathrm{mve}_j/\mathrm{bll}_j$$

then set

$$\mathrm{mve}_i = \mathrm{mve}_i + \mathrm{mve}_j, \quad \mathrm{bll}_i = \mathrm{bll}_i + \mathrm{bll}_j, \quad j = j - \mathrm{bll}_j$$

until $j < 1$. After this procedure the variables bll_i give the block lengths and mve_i give the block sums for equal m_i-s. (Notation bll comes from block length, mve from monotone averages.)

This is a dynamical programming solution of the problem. Starting with the trivial solution for one-element sequences we add step by step one element to the vector and amalgamate it into the last block as long as its average becomes large enough. Solving the problem in the same way for decreasing sequences and taking the minimum of the two sums of errors we get the monotone regression. The algorithm has two outputs: the monotone sequence m and an indication showing that increasing or decreasing sequences give the minimum. The number of steps needed in the algorithm is linear in p for all inputs t.

3.3. Monotone sum of squares

For any numbering π of the columns of the matrix A and any row $r \in \mathcal{R}$ first we form the sequence

$$t = (t_i = a(r, c_i),\ i = 1, \ldots, p)$$

from the elements of the row $r \in \mathcal{R}$ according to the numbering π and calculate the monotone regression m of the sequence t. For further reference we gather the corresponding monotone sequences m into the column monotone orderable matrix M_π and the *indicator of monotonicity,* which is a boolean function μ defined on $\mathcal{R}$: $\mu(r)$ if TRUE if m is increasing and FALSE otherwise. The sum of errors of monotone regression on rows of A according to the numbering π is denoted by $W_\pi(A)$. Given the matrix A this function is defined on the numberings of the columns of A. For the sake of simplicity here we denote the function by $w(\pi)$. The simplest way to minimize a combinatorial function is the use of genetical optimization [10], [13]. Without going into details we remark that in genetical optimization first we have to generate a subset of the set where the function to be optimized is defined. The subset is called population. We generate the starting population in the following way.

A natural aspirant for numbering the columns is the column average: let $\mathrm{ave}(c)$ be the average of the elements of the c-th column of matrix A. As it is easily seen, the numbering need not be restricted on integers, thus $\mathrm{ave}(c)$ is formally a permitted numbering. For any real valued numbering $f(c)$ let as define the corresponding integer valued numbering π by

$$f(c_1) < \cdots < f(c_p),$$

i.e. π is the numbering resulting such order of columns where the values of the function f are increasing. We shall use only functions having different values on

different columns. In genetical programming we use real valued numberings. The starting population is defined as

$$f(c) = \mathrm{ave}(c) + \kappa\sigma z(c),$$

where $z(c)$, $c \in \mathcal{C}$ are standard normal variables generated independently for each element of the population, κ is a parameter of the algorithm (we use $\kappa = 1/3$) and

$$\sigma = \sqrt{\frac{1}{np}\sum_{r,c}(a(r,c) - \mathrm{ave}(c))^2}.$$

Once a starting population is formed, the typical step in a possible version of the genetical optimization is the following. Let us choose two numberings π_1, π_2 from the population in such a way that $w(\pi_1) \le w(\pi_2)$. Next define the new real valued numbering $f(c)$ by coin tossing: for each column c let $f(c)$ be equal to $f_1(c)$ with probability 0.5 otherwise let it be $f_2(c)$ (f_i-s are the corresponding real valued numberings). In this step a certain mutation is also applicable: add if you want a random normal variable to $f(c)$ with zero expectation and with a variance appropriately chosen. The definite step of the algorithm is the substitution of the second numbering by the new one in case $w(\pi) < w(\pi_2)$. There are many natural stopping rules for the algorithm: we may terminate it either if the frequency of substitutions becomes small or if the difference between the maximum and minimum of $w(\pi)$ in the population is small. Nevertheless the simplest way to terminate the algorithm is counting the steps. After stopping the final result of the algorithm is the numbering π in the population minimizing the function $w(\pi)$.

3.4. Gradient methods

There are many natural perturbations of a numbering π, the most popular perhaps is the switch of two elements: choose two different columns c_1, c_2 and set $\pi^*(c_2) = \pi(c_1)$, $\pi^*(c_1) = \pi(c_2)$. We propose the following method of inserting instead this switch. Imagine, that there are p numbered places where the columns as guests may have a rest. Pick up one guest and move it to an arbitrary other place making empty place for it shifting the whole block between the new and old places with one step. Trying first all possibilities choose the perturbation resulting the best improvement in $w(\pi)$ if it has any, and stop the algorithm otherwise.

3.5. Multidimensional monotone curves

Let us fix an order for the rows, presently it has no importance what is the actual order. Each column c gives a point $D(c)$ in the n-dimensional space, the coordinates of $D(c)$ are the matrix elements in that column. What can we do with p points in the multidimensional space? Even in the multidimensional space we may figure out some version of monotone regression. A curve in the multidimensional

space is a mapping of the real line into the multidimensional space. We say that the curve is a *multidimensional monotone curve*, if the mapping is monotone in all coordinates. Let γ be an arbitrary monotone curve in the multidimensional space. Given the curve γ and the finite set of points $\mathcal{D} = \{D(c),\ c \in \mathcal{C}\}$ let us define the function

$$\Delta(\gamma, \mathcal{D}) = \sum_{c \in \mathcal{C}} d^2(\gamma, D(c)),$$

where $d(\gamma, D(c))$ is the Euclidean distance of the point $D(c)$ from the curve γ (which is the infimum of the distances between points of γ and the point $D(c)$). We can show that the infimum of $\Delta(\gamma, \mathcal{D})$ taken on all monotone curves equals to $W(A)$. This observation results the following algorithm.

Having a numbering of the columns we can build up a monotone curve in the multidimensional space in the following way. Let M_π be the monotone approximation of A according to ordering π. Let us drop the index π for a while and let us denote the points in the multidimensional space corresponding to M by $M(c)$, $c \in \mathcal{C}$. The numbering π gives a labelling c_i of the columns, let us label in the same way the points $M(c)$ and define the mapping γ from reals into the multidimensional space first on integers between 1 and p as $\gamma(i) = M(c_i)$, $i = 1, \ldots, p$. Next define the function $\gamma(t)$ for $i \leq t \leq i+1$, $1 \leq i < p$ by linear interpolation between points $\gamma(i)$ and $\gamma(i+1)$ and for other real numbers by linear extrapolation between points $\gamma(1)$ and $\gamma(p)$. This function is monotone.

The next step in the algorithm is to order numberings of columns to monotone curves γ. The whole algorithm consists of these two steps: to any numbering π we get a multidimensional monotone curve γ and to any multidimensional monotone curve γ we construct a new numbering π. The whole algorithm stops when the numbering does not change. Given a multidimensional monotone curve γ a real valued numbering of columns is provided by the real valued argument of the function parameterizing the monotone curve at closest point on to curve to the points corresponding to the matrix A.

3.6. Building up numberings

Given a numbering π of columns we can build up a new numbering in the following way. Let us apply a trivial dynamical application of the algorithm presented in the subsection 3.4. In that paragraph we found an optimal new value of $\pi(c)$ for any $c \in \mathcal{C}$. Now we extend the basic set of columns step by step following the ordering π.

3.7. McQueen's algorithm

In the previous sections we investigated the question of calculating $W(A)$ for a given matrix A, that means the numbering of the columns of A. Having a clustering ρ of the rows we apply this step in the centering step of McQueen's algorithm for the submatrices $A_i, 1 \leq i \leq k$, where k is the size of ρ.

Now we turn our attention to clustering rows. Given *any* finite set of numberings $\{\Pi : \pi\}$ of columns we can build up a clustering of rows in the assignment step of McQueen's algorithm in the following way: for any row $r \in \mathcal{R}$ let we choose the ordering minimizing $w_\pi(r)$ in $\pi \in \Pi$.

It may readily happen that some clusters become empty or they reach such a small number of elements what we count unnatural. This is the right time in the algorithm to reduce the set of clusters. We are planning to measure somehow the consistency of clusters, concluding in some cases that we ought to divide one cluster into parts. A natural way for doing that is just to reiterate the whole procedure recursively on the submatrix of the cluster. Presently we do not know how hard is the problem to determine $\mathcal{W}_k(A)$. Typically we get local minima in sense that our algorithms or some of them stop at a system of clusterings and numberings while the value of the risk function is not the best one found by different initializations. One specific feature of such situations is that applying many algorithms parallel they can improve situations after each other. It is a returning problem in clustering, what to do with a batch of different clusterings having quite good clustering power. How can we decide that in what sense are they different and one should like to use all of them wisely reaching somehow a unified solution. McQueen's algorithm provides a solution for the problem: let us form a pool from *all* numberings provided by the clusters and let us run the McQueen's algorithm step by step leaving only one of the numberings from the pool. In course of this algorithm we determine the smallest increase in the penalty function of clustering (here it is the sum of monotone sum of squares of submatrices A_i formed by the clustering). We drop the numbering giving the smallest increase and reiterate the step as long as the number of clusters reaches the wanted level.

For detecting the interrelations of clusters the following matrix is useful. The *matrix of distances of clusters* is a square matrix having k rows and columns. The element of the matrix in the i-th row and j-th column is the square average of the monotone errors of rows belonging to the i-th cluster calculated with respect to the numbering of the j-th cluster. Let us observe that these errors are calculated in course of McQueen's algorithm. We use the spanning tree presented by this matrix parallel with the spanning tree given by directed monotone distances of numberings of the clusters. Given any directed monotone distance on a finite set, the *spanning tree* is a tree having the points of the set as vertices and minimizing the sum of directed distances on edges appropriately directed.

3.8. Initializations

How can we build up a new clustering? Following the lines of monotone sum of squares we may define a new function serving as some distance between points of multidimensional space. This is the *directed monotone distance* defined in the following way. Let t and u be two row vectors of A, they are points in the multidimensional space. The directed monotone distance of t and u is defined only in case when the coordinates of t are not constant, i.e. there are differences among the coordinates of t. (Let us remark, that constant rows may be eliminated from A as a first step.) Given the coordinates of u, the *rank numbers* of u are given by a mapping ψ from the reals onto the positive integers such that (remember that u is a p-dimensional vector having coordinates $(u(c), c \in \mathcal{C})$)

(i) for all pairs c and c' in $\mathcal{C}$ if $u(c) < u(c')$, then $\psi(u(c)) < \psi(u(c'))$,
(ii) the set $\{\psi(u(c)),\ c \in \mathcal{C}\}$ is an interval of integers containing the number 1 as an element.

The numbers

$$\{\psi(u(c)),\ c \in \mathcal{C}\}$$

are the rank numbers of u. The largest $\psi(u(c))$ equals the number of different values among the coordinates $u(c)$, $c \in \mathcal{C}$. The *extended monotone regression* of the vector t with respect to u is the vector m having the same dimension and the same rank numbers as u and minimizing

$$\sum_{c \in \mathcal{C}} (t(c) - m(c))^2.$$

Let us denote the minimum by $\mathrm{EMR}(t \mid u)$. Directed monotone distance $d(t \mid u)$ is defined in the following way:

$$d(t \mid u) = \frac{\mathrm{EMR}(t \mid u)}{\sum_{c \in \mathcal{C}} (t(c) - t_0)^2},$$

where t_0 is the average of the numbers $t(c)$, $c \in \mathcal{C}$.

Given the directed monotone distance $d(t \mid u)$ we use the spanning tree defined by the distance of rows. There are many algorithms for constructing a spanning tree, in one of them edges are added to a forest greedily choosing step by step the pair having minimal distance and keeping the forest property. In this algorithm we can stop when the number of trees equals to k, and forming clusters from trees.

3.9. The main algorithm

Having the following three algorithms:

(i) initialization for a finite set Π of orderings
(ii) assignment: calculating a clustering ρ to any set Π of orderings
(iii) centering: calculating the orderings $\pi_i, 1 \le i \le k$ to any clusterings ρ of size k

the general algorithm is the following: start with (i) and iterate (ii) and (iii) until stopping. The only formal step needed here serving as a junction between (ii) and (iii) is to form a set Π from a vectors of orderings π_i, $1 \le i \le k$ onto Π. Being in a finite setup the stopping rule is fixation.

3.10. Confidence intervals

A possible statistical model of the algorithms presented here is the following. We have the following ingredients:

(i) a clustering ρ of rows with size k
(ii) a numbering π_i of columns for all $1 \le i \le k$ such that for all positive integers j between 1 and p the number j is taken exactly once by π_i
(iii) a positive real number σ giving the standard error of experiments

$$a(r,c), \quad r \in \mathcal{R},\ c \in \mathcal{C}$$

(iv) a row monotone increasing matrix B indexed by $\mathcal{R}$ as row index but with integers between 1 and p as column index: $b(r,i) \le b(r,i+1)$ for all $r \in \mathcal{R}$ and $1 \le i < p$.

With these ingredients the stochastic model is defined as

$$a(r,c) = \varepsilon(r) b\big(r, \pi_{\rho(r)}(c)\big) + \sigma z(r,c),$$

where the random variables $\varepsilon(r)$, $r \in \mathcal{R}$ are iid standard ± 1 and the variables $z(r,c)$, $r \in \mathcal{R}$, $c \in \mathcal{C}$ are independent standard normals, and σ is a positive constant.

Let us say that the row monotone increasing matrix B is blocked with respect to the clustering ρ of size k of rows if for all $1 \le i \le k$ there is a partition $\mathcal{P}_i$ of integers between 1 and p into consecutive intervals having the following property: all numbers $b(r,i)$, $b(r,j)$ are equal if i and j belong to the same element of the partition $\mathcal{P}_{\rho(r)}$. For such cases we should like to detect the block structure of the theoretical matrix B according to the basic rules of statistical confidence intervals; which is to minimize the probability of events

(i) declaring difference in cases there is equality
(ii) declaring equality in cases there is difference.

Our solution is the following. We evaluate a system of partitions $\mathcal{P}_i$, $1 \le i \le k$ using the increases in monotone sums caused by forcing equalities in extended monotone regressions dictated by $\mathcal{P}_i$, $1 \le i \le k$. For a predetermined level of increase we determine the largest possible partitions.

3.11. Preprocessing matrices

Decomposition clusters of monotone orderings formally may be used for any real matrix but some preprocessing may be advisable. The algorithm is powerful on matrices having *equivalent* columns in the sense that the rank numbers in rows are close to random, all permutations may appear among them. In case the scale used in different columns is different, the algorithm may miss the point. A general preprocessing is the quantile transformation for the columns, or any of it's version. The simplest solution is just to transform linearly all columns into interval $(0, 1)$.

4. Decomposition clusters of partitions

4.1. Ordered partitions

Let $\mathcal{C}$ be a finite set of p elements and let q be a positive integer such that $q \le p$. The function

$$\beta : \mathcal{C} \to \{1, \dots, q\},$$

is said to be an *ordered partition* if all integers between 1 and q appear at least once among the $\beta(c)$-s. We say that the number q is the size of β. Two ordered partitions α and β are different, if there is an element c of $\mathcal{C}$ such that $\alpha(c) \ne \beta(c)$. We shall denote by $\mathcal{B}_p$ the set of all ordered partitions of p elements, and by G_p the number of elements of $\mathcal{B}_p$. An easy recursion may help to evaluate G_p:

$$G_p(q) = \sum_{s=1}^{p-q+1} \binom{p}{s} G_{p-s}(q-1),$$

where $G_p(q)$ is then number of elements in $\mathcal{B}_p$ with size q (especially $G_p(1) = 1$, $G_p(p) = p!$).

4.2. Normal variables

Let β be an ordered partition of the set $\mathcal{C}$. We say that the random variables $Z = (z(c),\ c \in \mathcal{C})$ follow the *ordered partition normal law according to* β if the $z(c)$-s are independent normal variables with expectation $Ez(c) = \mu(\beta(c))$ and with variance σ, where

$$\mu(1) < \mu(2) < \cdots < \mu(q),$$

and q is the size of β.

A possible estimate of β from Z is the following. We fix two positive constants u, v called *center* and *range*. Let us determine for each $1 \le s \le p$ the statistics

$$Q_s = \min_{t_1,\dots,t_s} \sum_{c\in\mathcal{C}} \min_j (z(c) - t_j)^2.$$

Let κ be determined by

$$\kappa = \min\left\{ s : \left| \frac{Q_s}{p+1-s} - u\sigma^2 \right| < v\sigma^2 \right\},$$

if there is any k with the required property, otherwise we define κ by

$$\kappa = \min\left\{ s : \frac{Q_s}{p+1-s} - u\sigma^2 < v\sigma^2 \right\}.$$

(Observe that $Q_p = 0$.)

4.3. Simultaneous coloring

A coloring of a finite set is an *unordered* numbering of the elements of the set. *Simultaneous coloring of the set of rows and columns of a matrix* is a pair $(\rho(r), \beta(c))$ of colorings of the sets $(\mathcal{R}, \mathcal{C})$ having the following properties:

(i) there are integers $k \ge 1$, $q \ge 1$ such that $1 \le \rho(r) \le k$ for all $r \in \mathcal{R}$, $1 \le \beta(c) \le q$ for all $c \in \mathcal{C}$

(ii) all integers between 1 and k appear at least once among $\rho(r)$-s and all integers between 1 and q appear at least once among $\beta(c)$-s.

Two simultaneous colorings are equivalent if they give the same partitions of the sets $\mathcal{R}$, $\mathcal{C}$ respectively. We refer to the numbers k, q as the *sizes* of the colorings. In a seminal paper [27] Szemerédi proved a structural property of graphs: the vertices of all graphs have coloring such that the edges joining vertices colored with a given pair of colors behave as they were random. We refer to the papers [4], [8], [11], [19] showing the effect of Szemeredi's regularity lemma in graph theory. Rephrasing Szemerédi's random graph model for real matrices we get the following stochastic model:

$$a(r,c) = b\left(\rho(r), \beta(c)\right) + \sigma\left(\rho(r), \beta(c)\right) z(r,c), \quad r \in \mathcal{R},\ c \in \mathcal{C},$$

where

(i) the pair (ρ, β) is a simultaneous coloring of $\mathcal{R}$, $\mathcal{C}$ with sizes k, q

(ii) the matrices $(b(i,j),\ 1 \le i \le k,\ 1 \le j \le q)$, $(\sigma(i,j),\ 1 \le i \le k,\ 1 \le j \le q)$ are arbitrary real valued matrices

(iii) the elements of the random matrix $(z(r,c),\ r \in \mathcal{R},\ c \in \mathcal{C})$ are independent standard normal.

Actually in Szemerédi's theorem the distribution of the elements of the submatrices $A_{ij} = (a(r,c),\ \rho(r) = i,\ \beta(c) = j)$ is arbitrary but for our aim the reduction to Gaussian distribution is admissible.

We say that the matrix A has a *noiseless Szemerédi structure*, if σ is zero, and denote by $\mathcal{N}(k,q)$ the set of all matrices having noiseless Szemerédi structure with coloring sizes k, q. Singular value decomposition applies in finding the colorings: let $B = VD$ the singular value decomposition of $B = (b(i,j),\ 1 \le i \le k,\ 1 \le j \le q)$, then

$$b(i,j) = (v_i, d_j),$$

where $(\cdot,\cdot)$ denotes the scalar product, $(v_1, \ldots, v_k)$, $(d_1, \ldots, d_q)$ are s-dimensional vectors, and s is the rank of B. From $a(r,c) = b(\rho(r), \beta(c))$ it follows that the *same* vectors provide singular value decomposition of A, if it is noiseless. Exposed to noise, these vectors may be found by simultaneous clustering of vectors in the singular value decomposition of A.

We applied spectral methods for graph clustering in [3], the idea to apply spectral methods for Szemeredi theorem if discussed in [2]. Biclustering and co-clustering is widely used for microarrays [7], the method is originated by [15]. The main difference between the two models is that in biclustering only homogeneous submatrices are picked out and the whole checkerboard structure is not used yet. But the two theories are converging, the argumentation of [18] is very close to our presentation.

4.4. Partition clusterings

There is no need to use the same partition β for all row clusters in Szemerédi's model. Furthermore for a fixed column partition we may drop the condition of the model that the expected values in one subset of the partition are the same in different rows. The first person performing clustering was Cinderella, who was obliged to decompose the mixture of thousands of seeds. A computerized Cinderella has a data matrix: the columns represent the seeds thus her matrix has thousands of columns. The rows represent the different parameters of the seeds. Cinderella collected the physical characteristics of the seeds in the first 12 rows and in other 72 rows she listed the concentration of different types of proteins in the seeds. Using the first 12 rows she got the well known types of seeds: corn, rye, wheat,..., etc. but the representation of the seeds in protein space led to completely different clusters. Working on her PhD thesis, Cinderella wanted to describe the two different

clusterings. But her Wicked Step Mother mixed the rows, too. Thus she forced Cinderella to discover partition clustering.

Let $\rho(r)$, $r \in \mathcal{R}$ be a coloring of the rows with size k and let $\beta_1, \dots, \beta_k$ be *arbitrary* colorings of columns. The sum of square errors of the system is defined by $\sum_{i=1}^{k} \mathcal{G}(A_i)$, where A_i is the submatrix formed from rows $\rho(r) = i$ and

$$\mathcal{G}(A_i) = \sum_{r\,:\,\rho(r)=i} \sum_{c\in\mathcal{C}} \left(a(r,c) - b_i(r,\beta_i(c))\right)^2,$$

where

$$b_i(r,j) = \frac{\sum_{c:\beta_i(c)=j} a(r,c)}{\sum_{c:\beta_i(c)=j} 1}.$$

Colorings or unordered partitions are clusterings. In partition clustering the columns of the submatrix A_i are clustered into q_i clusters as multidimensional points according to β_i around the centrums $(b(i,j),\ j = 1, \dots, q_i)$, where q_i is the size of β_i. The row clustering ρ is using the column clusterings β_i as cluster centers: in the assignment step of McQueen's algorithm for each row $r \in \mathcal{R}$ we can choose the best partition minimizing the quantity

$$g_i(r) = \sum_{c\in\mathcal{C}} (a(r,c) - b_i(r,\beta_i(c))^2.$$

We shall denote the partition of columns given by β_i with $\mathcal{P}_i$. All details of the calculations are easily extendable to the general case whenever we are given the system $\mathcal{P}_1, \dots, \mathcal{P}_k$ of partitions of columns. One possible source for such a system is the experimental design: they are the designers of microarray measurements who can tell to statisticians how to form partitions from conditions. Nevertheless there are statistical methods helping the search of partitions. Namely it is the likelihood function which may direct the search: systems with good likelihood are statistically more acceptable than those with poor likelihood. Some caution is, however needed: partitions with large size have to penalized for counterbalancing their power in likelihood.

4.5. Row couplings

Let us construct for any pairs $r, s \in \mathcal{R}$ the set $Q(r,s)$ of two-dimensional points Q_c, $c \in \mathcal{C}$ having coordinates $(a(r,c), a(s,c))$. For pairs $r, s : \rho(r) = \rho(s) = i$ the set $Q(r,s)$ is clustered into q_i clusters where q_i is the size of β_i, otherwise it has some checkerboard structure formed by the two involved clusters. A statistic capturing cluster pairs is

$$K(r,s) = \sum_{c\in\mathcal{C}} \sum_{i=1}^{t} d\big(Q_c, Q_{c(i)}\big),$$

where $t \geq 1$ is a parameter of the statistic, d is the Euclidean distance, and $Q_{c(i)}$ is the i-th nearest neighbor of Q_c in $Q(r,s)$. The goodness of a clustering ρ of the rows is quantified by

$$\mathcal{Q} = \sum_{\rho(r)=\rho(s)} K(r,s).$$

For fixed size k we can determine the minimizing clustering by genetic optimization.

4.6. Finding hidden structure in a large data set

Let us consider a sample where p-dimensional random variables from mixtures are observed. The different mixtures are defined with the help of given partitions defined on the set $\{1,\dots,p\}$. Knowing the partitions, we can use the information about known connections among coordinates of the multivariate observations. It is supposed that independent uniform random variables are belonging to each set of a partition, further an observed variable is a function of these hidden variables and a p-dimensional normal random variable.

The values of a p-dimensional random variable $\mathbf{Y}$ are observed and $\mathbf{Y}$ is given as

$$\mathbf{Y} = \mathbf{V}^{(\delta)} + \mathbf{Z},$$

where $\mathbf{Z}$ is a p-dimensional normal random variable with independent coordinates, 0 means and unknown standard deviations σ_j, $j = 1,\dots,p$, δ is a random variable with integer values $1,\dots,k$ and $P(\delta = i) = p_i$, $\sum_{i=1}^{k} p_i = 1$. $\mathbf{V}^{(\delta)}$ and $\mathbf{Z}$ are independent random variables. $\mathbf{V}^{(\delta)}$ is a mixture of random variables $\mathbf{V}^{(i)}$, $(i = 1,\dots,k)$ and they are independent of δ. To define the random variables $\mathbf{V}^{(i)}$ $(i = 1,\dots,k)$ some further notations are needed. For each i $(i = 1,\dots,k)$ there is given a partition $(\beta_{i,1},\dots,\beta_{i,p})$ with partition size q_i.

Independent uniform $[0,1]$ random variables $U_j^{(i)}$ $(j = 1,\dots,q_i,\ i = 1,\dots,k)$ are assigned to the partitions. The p-dimensional random variable $\mathbf{V}^{(i)}$ is defined as

$$\mathbf{V}^{(i)} = \left(g\left(U^{(i)}_{\beta_{i,1}}, \Theta_1\right), \dots, g\left(U^{(i)}_{\beta_{i,p}}, \Theta_p\right)\right), \tag{1}$$

where $g(u,\Theta)$ is a given function and $\Theta_1,\dots,\Theta_p$ are unknown multidimensional parameter vectors.

A simple example is when only two partitions are given, one is $(1,\dots,1)$ and the other one is $(1,2,\dots,p)$. Then the random variable δ has two values. When $\delta = 1$ each coordinate of $\mathbf{V}^{(1)}$ depends on the same uniform variable and when $\delta = 2$ the coordinates of $\mathbf{V}^{(2)}$ are depending on p independent uniform variables.

Let us remark that using uniform random variables is not a restriction of the model since a quantile function of an arbitrary distribution can be incorporated in function g.

The origin of the model is the attempt to assess microarray data. In microarrays thousands of gene expressions are measured under different conditions. Considering one gene, suppose that its expression level is measured under p different conditions. An example is measuring gene expression levels during the life of an organism at p different time. Other examples are when gene expression levels are considered in case-control studies. If the expression levels of a gene are not affected by conditions then they are referred as equally expressed, otherwise they are differentially expressed. Partitions of the p conditions are given according to the experiment. In this application the partition ($\beta_{i,j} = 1$, $j = 1, \dots p$) is always a possibility, since most genes are expressed identically under different conditions.

4.7. Estimating the hidden variables

We present a method to estimate the parameters and the hidden variables of the model. At first we estimate the model parameters with the help of the maximum likelihood method and then we use conditional expectation to estimate the values of the hidden uniform variables. Knowing the parameters of the model for each observation the probability that it belongs to any of the k partitions can be estimated using the Bayes' rule. Observations can be classified according to the most probable partitions.

Suppose that it is given a sample of n elements $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$ from the above model with k previously specified partitions. Set $\mathbf{y}_\ell = (y_{\ell,1}, \dots, y_{\ell,p})$, $\ell = 1, \dots, n$. Our goal is to estimate the parameters Θ_j, σ_j, $(j = 1, \dots, p)$, p_i, $(i = 1, \dots, k)$ and the values of hidden variables.

The sample comes from a mixture of distributions with likelihood function

$$\prod_{\ell=1}^{n} \left(\sum_{i=1}^{k} p_i f_i(\mathbf{y}_\ell) \right) \tag{2}$$

where $f_i(\mathbf{y}_\ell)$ is the density function of $\mathbf{V}^{(i)}$ defined in (1). It is easy to see that

$$f_i(\mathbf{y}_\ell) = \prod_{t=1}^{q_i} \int_0^1 \prod_{j:\beta_{i,j}=t} \frac{1}{\sigma_j} \varphi \left(\frac{y_{\ell,j} - g(u, \Theta_j)}{\sigma_j} \right) du, \tag{3}$$

where φ denotes the density function of a standard normal random variable. The maximum likelihood method can be used to estimate the model parameters. Using the estimated model parameters Θ_j, σ_j, p_i, $(j = 1, \dots, p,\ i = 1, \dots, k)$ for each $\mathbf{y}_\ell$, $\ell = 1, \dots, n$ it is possible to estimate the posterior probability that the corresponding hidden variable from the i-th partition as

$$p_{\ell,i} = P(\delta = i \mid \mathbf{Y} = \mathbf{y}_\ell) = \frac{p_i f_i(\mathbf{y}_\ell)}{\sum_{m=1}^{k} p_m f_m(\mathbf{y}_\ell)} \qquad i = 1, \dots k.$$

Conditional expectation can be used to estimate the value of the hidden variable.

THEOREM. *For $\ell = 1, \dots, n$, $j = 1, \dots, p$*

$$\hat{U}_{\ell,j} = E\left(U_j^{(\delta)} \mid \mathbf{Y}_\ell = \mathbf{y}_\ell\right) = \sum_{i=1}^{k} p_{\ell,i} \left(\frac{\int_0^1 u \prod\limits_{m:\ \beta_{i,m}=\beta i,j} \varphi\left(\frac{y_{\ell,m}-g(u,\Theta_m)}{\sigma_m}\right) du}{\int_0^1 \prod\limits_{m:\ \beta_{i,m}=\beta i,j} \varphi\left(\frac{y_{\ell,m}-g(u,\Theta_m)}{\sigma_m}\right) du} \right).$$

The proof of the theorem is based on the following simple lemma:

LEMMA. *Let $\xi, \eta_1, \dots, \eta_n$ be independent random variables with continuous density functions $f_\xi, f_{\eta_1}, \dots, f_{\eta_n}$, respectively. Let $g_1, \dots, g_n : \mathbf{R} \to \mathbf{R}$ be continuous functions. Then*

$$\begin{aligned} &E(\xi \mid g_1(\xi) = \eta_1, \dots, g_n(\xi) = \eta_n) \\ &\qquad = \frac{\int_{-\infty}^{\infty} u\, f_\xi(u)\, f_{\eta_1}(g_1(u)) \dots f_{\eta_n}(g_n(u))\, du}{\int_{-\infty}^{\infty} f_\xi(u)\, f_{\eta_1}(g_1(u)) \dots f_{\eta_n}(g_n(u))\, du}. \end{aligned} \tag{4}$$

5. Concluding remarks

Let $A = (a(r,c),\ r \in \mathcal{R},\ c \in \mathcal{C})$ be an arbitrary data matrix, and let us denote its row vectors by Y_c. Let us denote by $\mathcal{T}_h(A)$ the minimum of the travelling cost $C(T)$ defined in subsection 2.2. It depends on A and h (in Section 2 we identified $\mathcal{C}$ with $\{1, \dots, p\}$). In subsection 3.1 we defined $\mathcal{W}_k(A)$ as the minimum of the sum of monotone sum of squares of row clusterings into k clusters. In the subsection 4.3 $\mathcal{N}(k,q)$ was the set of noiseless matrices: let us denote now by $\mathcal{S}_{kq}(A)$ the minimum of sum of square errors of the approximating noiseless matrices. The three statistics $\mathcal{T}_h(A)$, $\mathcal{W}_k(A)$ and $\mathcal{S}_{kq}(A)$ quantify the goodness of fit of our different clustering methods. We have discussed already some goodness-of-fit procedures of the models leading to these statistics. In a certain sense clustering data matrices is pattern recognition: the structure of the clusters are the patterns to be identified. In microarray measurements the whole set of possible questions is determined by nature but the set of answers, the set of conditions under what the gene expressions are measured is unto the design of the experimenters. In a good design the structure of the expected answers is determined by the design itself. The general way of model checking is error analysis: to test whether the deviations in the data follow the noise structure imprinted in the model. In case of DCP and DCM we have an adhoc method, the coupling of rows, which was originally a test for DCP and led us to DCM. In DCP for rows belonging to the same cluster the coupled picture itself shows clusters, where columns belonging to the same subset of the partition form one cluster. It was our expectation to see in our data the clusters but we have seen instead the monotone curves of DCM.

References

[1] L. G. Balázs, Solution of the basic equation of stellar statistics, *Inverse Problems* **11** (1995), 731–741.

[2] M. Bolla, Recognizing linear structure in noisy matrices, *Linear Algebra and its Applications*, (2005).

[3] M. Bolla and G. Tusnády, Spectra and optimal partitions of weighted graphs, *Discrete Mathematics* **128** (1994), 1–20.

[4] F. R. K. Chung, Regularity lemmas for hypergraphs and quasi-randomness, *Random Structures and Algorithms* **2** (1991), 241–252.

[5] L. A. Cogburn, W. Wang, W. Carre, L. Rejtő, T. E. Porter, S. E. Aggrey and J. Simon, System-wide chicken DNA microarrays, gene expression profiling, and discovery of functional genes, *Poultry Science Association* **82** (2003), 939–951.

[6] L. A. Cogburn, X. Wang, W. Carre, L. Rejtő, S. E. Aggrey, M. J. Duclos, J. Simon and T. E. Porter, Functional genomics in chickens: development of integrated-systems microarrays for transcriptional profiling and discovery of regulatory pathways, *Comparative and Functional Genomics* **5** (2004), 253–261.

[7] Y. Cheng and G. M. Church, Biclustering of expression data, *Proceedings of ISMB*, AAAI Press, 2000, 93–103.

[8] A. Czygrinow and V. Rödl, An algorithmic regularity lemma for hypergraphs, *SIAM J. Comput.* **30** (2000), 1041–1066.

[9] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, Clustering large graphs via the singular value decomposition, *Machine Learning* **56** (2004), 9–33.

[10] S. Forrest, Genetic algorithms: principles of natural selection applied to computation, *Science* **261** (1993), 872–878.

[11] P. Frankl and V. Rödl, The uniformity lemma for hypergraphs, *Graphs Combin.* **8** (1992), 309–312.

[12] A. Frieze and R. Kannan, Quick approximation to matrices and applications, *Combinatorica* **19** (1999), 175–220.

[13] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, 1989.

[14] G. Halász, Personal communication, 1984.

[15] J. A. Hartigan, Direct clustering of a data matrix, *Journal of the American Statistical Association* **67** (1972), 123–129.

[16] S. Kauffman, *The origins of order, self-organization and selection in evolution*, Oxford University Press, New York, 1993.

[17] S. Kauffman, C. Peterson, B. Samuelsson and C. Troein, Genetic networks with canalyzing Boolean rules are always stable, *Proceedings of the National Academy of Sciences of the United States of America* **101** (2004), 17102–17107.

[18] Y. Kluger, R. Basri, J. T. Chang and M. Gerstein, Spectral biclustering of micoarray data: Coclustering genes and conditions, *Genome Research* **13** (2003), 703–716.

[19] J. Komlós and M. Simonovits, Szemerédi's regularity lemma and its applications in graph theory, *Combinatorics, Paul Erdős is eighty*, Vol. 2, (Keszthely, 1993), 295–352, Bolyai Soc. Math. Stud., 2, János Bolyai Math. Soc., Budapest, 1996.

[20] G. Parmigiani, E. S. Garrett, R. A. Irrizary and S. L. Zeger, *The analysis of gene expression data, methods and software*, Springer, 2003

[21] A. Perczel, M. Hollósi, G. Tusnády and G. D. Fasman, Convex constraint analysis: a natural deconvolution of circular dichroism curves of proteins, *Protein Engineering* **4** (1991), 669–679.
[22] R. Pick and G. Tusnády, Decomposition of mixtures, *Studia Scientiarum Mathematicarum Hungarica* **15** (1980), 31–37.
[23] L. Rejtő and G. Tusnády, Evolution of random Boolean NK-models in Tierra environment, *Limit theorems in probability and statistics,* Vol. II, (ed. by I. Berkes, E. Csáki and M. Csörgő), Budapest, 2002, 499–526.
[24] L. Rejtő and G. Tusnády, Reconstruction of Kauffman networks applying trees, Manuscript, Submitted for publication, 2004.
[25] M. Schena, *Microarray analysis*, Wiley-Lis, 2003.
[26] T. Speed, *Statistical analysys of gene expression microarray data*, Chapman & Hall, 2003.
[27] E. Szemerédi, Regular partitions of graphs, Colloques Internationaux C.N.R.S. No. 260, *Problèmes combinatoires et théories des draphes*, Orsay, 1976, 399–401.
[28] C. Zhang, Fourier methods for testing mixing densities and distributions, *The Annals of Statistics*, **18** (1990), 806–831.

(Received: January 19, 2005)

Lídia Rejtő
Alfréd Rényi Mathematical Institute of the Hungarian Academy of Sciences
H-1364 Budapest, P.O. Box 127
Hungary
and
University of Delaware
Statistics Program
214 Townsend Hall, Newark, DE 19717-1303
USA

Gábor Tusnády
Alfréd Rényi Mathematical Institute of the Hungarian Academy of Sciences
H-1364 Budapest, P.O. Box 127
Hungary