file: lb0ff.tex, May 2, 2004

Kohonen's self-organizing maps as applied to graphical visualization of some yeast DNA data

Anna Bartkowiak[1], Adam Szustalewicz[1]
Stanisław Cebrat[2], Paweł Mackiewicz[2]
[1] Institute of Computer Science, University of Wrocław
Przesmyckiego 20, 51–151 Wrocław, Pl [2]Institute of Genetics and Microbiology,
University of Wrocław Przybyszewskiego 63/77, 51–148 Wrocław, Pl

Abstract

We analyze a set of data describing N=3300 yeast genes, each gene characterized by d=13 variables (traits). First we performed an explorative data analysis and stated a high multivariate kurtosis. Next we clustered the data and visualized them using Kohonen's self-organizing maps. This permitted us to get an idea how the data are distributed in the multivariate space.

Keywords: explorative analysis, clustering, self-organizing maps, yeast genome.

1 Genesis and aim of the paper

The DNA code is a great challenge to contemporary research, especially to biochemistry and genetics, and also to the information theory and data analysis in general.

For several years the first two authors have been watching the investigations of the yeast genome conducted in the Dept. of Genomics, Inst. of Genetics and Microbiology, Univ. of Wrocław. The DNA code is really fascinating. Seemingly simple, it bears unimaginable number of possibilities, and is extremely attractive from the point of data analysis.

The researchers from the mentioned Department of Genomics succeeded to make a nontrivial conversion from a linear sequence of basic nucleotides to a multivariate vector of data characterized by 13 variables. In such a way we have been transferred to the firm ground of multivariate analysis.

Now, in this paper, we would like to look at the genes as at a set of data vectors, or – equivalently – to look at the genes as at a multivariate data cloud located in a 13–dimensional data space. What can be said about this data set? Is it normally distributed? Can we visualize it in a satisfactory manner in a plane?

The paper is scheduled as follows: In Section 2 we present in more detail the derivation of the data and the meaning of the established variables. Section 3 shows an exploratory analysis of the data. This is done by drawing a scatter-plot matrix of selected pairs of variables and plotting Mahalanobis distances. Also a search of outliers and tests on multivariate normality of the data are carried out. The multivariate normality is decidedly rejected. In Section 4 we perform a visualization of the data using Kohonen's self-organizing maps. The method performs a kind of clustering with preserving the topology of the data. Section 5 shows the projection of a set of 'non-genes' onto the map constructed from the genes. One may see that the distributions of these two groups of data are somehow shifted.

2 The data

The ‘life’ information of a cell is hidden in chromosomes. Speaking more precisely, this information is coded in DNA (deoxyribonucleic acid) sequences composed of four bases (nucleotides) denoted by A, G, C, T. This is the alphabet of the code. The code is organized in triplets called codons.

There are $3^4 = 64$ possible codons, of which 61 (including start codon) code for amino acids. The three remaining codons code for stop - which means the end of the coding sequence. A sequence of codons beginning with a start codon and ending with a stop codon is called an ORF (Open Reading Frame). The sequence delimited in such way may contain some specific vital information for the organism. In such a case, using a popular (non–technical) language, it is referred to as a ’gene’. To identify a ’gene’, we first need to find a delimited sequence, i.e. an ORF, and next to find out what its function is. Recognizing the function coded by an ORF is a laborious task and for part of the ORFs their vital functions are unknown and we do not know whether they have a vital function at all. Summarizing, it can be said that every ’gene’ is an ORF, however not every ORF is a ’gene’.

The statistical issues of the code have been considered for a long time by the researchers. A review of the problems – from the point of mathematical statistics – may be found, e.g., in Braun and Müller (1998). Some specific topics, to mention a few out of many published in statistical journals, were considered by Avery and Henderson (1999), Muri (1998), Prum *et al.* (1995), Kamb *et al.* (1995).

A completely different approach was proposed by Cebrat et al. (1997, 1998). They were concerned with the yeast genome *Sacharomyces cerevisiae*. The yeast genome contains 16 chromosomes, comprising sequences totalling about 13 millions bases.

The mentioned authors proposed to characterize the DNA sequences constituting each ORF by a *spider-plot*. Next they have described each spider-plot by 13 characteristics called in the following ’traits’ or ’variables’. In such a way our feet were put on the firm ground of multivariate data analysis.

An exemplary spider-plot is shown in Figure 1.

A spider plot is obtained by performing a specific DNA walk (see Cebrat and Dudek, 1998, Cebrat et. al. 1997, 1998). An imaginary walker proceeds to visit the first nucleotides of the codons, starting from the begin of the ORF. Simultaneously, correspondingly to the bases appearing in the visited places of the codons, a graph is constructed using a marker. Starting from the (0,0) point of the graph, we move the position of the marker a unit up if the nucleotide is G, down if it is C, right if it is A and left if it is T. This is done until the walker reaches the end of the ORF. The same is done, when the walker visits the second and third position of the codons in the given ORF. In such a way 3 independent graphs are obtained; they reflect the sequential occurrence of bases in the 1st, 2nd and 3rd position of the codon. All the 3 graphs are put together in one plot, called spider-plot. The planar plot exhibits a structure which resembles a spider with 3 ’legs’. The ’legs’ correspond to three positions in the codons visited by the walker.

Constructed in such a way, a spider-plot expresses the prevalence of some nucleotides in the 1st, 2nd and 3rd position of the codons occurring in the analyzed ORF.

The leg positioned in the first quadrant of the plane, obtained when visiting the first places of the codons, expresses the prevalence of occurrence of the A and the G nucleotides (bases: Adenine and Guanine, described jointly as purines) – over the T and the C nu-

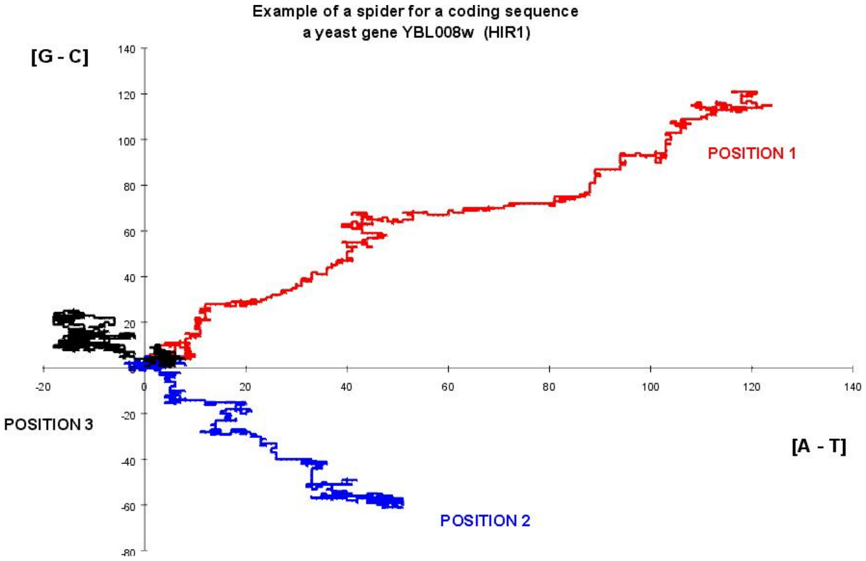


Figure 1 *Spider plot for the yeast gene YBL008w (HIR1); position 1 - the walker visits every first nucleotide of codons and moves a unit up if the nucleotide is G, down if it is C, right if it is A and left if it is T; position 2 - the walker visits every second nucleotide of codons, and proceeds the same way; position 3 - the walker visits every third nucleotide of codons, and proceeds the same way.*

cleotides (bases: Thymine and Cytosine, described jointly as pyrimidines). Similarly, the leg positioned in the 2nd quadrant of the (x,y) plane expresses the prevalence of the A and the C nucleotides (bases) – when observing the 2nd positions of the codons. Finally, the 3rd leg positioned in the 3rd or 4th quadrant expresses the prevalence of the T base over the A base in the third positions of the codons.

Our analysis will be based on a data set comprising 6330 yeast ORFs downloaded from http://www.mips.biochem.mpg.de in 2000. A total of $N = 3300$ genes (ORFs with assigned function) was then known. The ORFs were of variable length (l_i). Since the ORF is composed from triplets of bases, the number l_i should be a multiple of '3': $l_i = 3 * k_i,\ i = 1, \ldots, N$.

From each ORF (i.e., from each DNA sequence representing one ORF) a spider-plot was constructed.

It was interesting to state that for most of the investigated 3300 genes:

- the first positions of the codons yielded a 'leg' positioned in the first quadrant of the (x,y) plane,
- the second positions of the codons yielded a 'leg' positioned in the second quadrant of the (x,y) plot (counting clockwise),
- the third positions of codons yielded usually an irregular line, clumped either in one or both of the 3rd and the 4th quadrant.

Each spider-plot was then converted to a vector of 13 characteristics, constituting the data vector for the given gene. Detailed information how to define the characteristics may be found at http://smorfland.microb.uni.wroc.pl.

The derived 13 characteristics have the following meaning

V1, V2, V3 – angle1, angle2, angle3; angles between the 'legs' and the x-axis,
V4, V5 – (x1, y1), coordinates of the extreme point of the first 'leg',
V6 , V7 – (x2, y2), coordinates of the extreme point of the second 'leg',
V8, V9 – (x3, y3), coordinates of the extreme point of the third 'leg',
V10 – (l_i), total length of the DNA sequence identified with the given ORF,
V11, V12, V13 – rho1, rho2, rho3, normalized lengths of the 1st, 2nd, and 3rd leg.

Thus, to obtain a data table of size 3300 × 13, a total of $N = 3300$ spider-plots (one for each gene) had to be constructed and converted to corresponding data vectors[1].

In such a way we got the data table named GENES with $N = 3300$ rows and $d = 13$ columns. Each row represents one gene, which is now characterized by 13 variables.

In a similar way the data table named NON-GENES was obtained. It was derived from ORFs, which in 2000 were not recognized as 'genes'. There were 3030 such ORFs.

Thus, in the following, when speaking about 'genes', we will have in mind ORFs, which in 2000 were *recognized genes*, i.e. with recognized life functions; when speaking about 'non-genes', we will have in mind ORFs, whose life functions were not exactly known in year 2000. Therefore, the set of NON-GENES may contain both not-recognized-yet genes and some ORFs without any life function.

3 Exploratory analysis of the data set

Pair-wise scatter-plots of the variables angle1, angle2, angle3, rho1, rho2, rho3 and length, crossed with the variables x1, y1, x2, y2, x3, y3, are shown in Figure 2.

Looking at the scatter-plots shown in Figure 2 one may observe peculiar shapes in them. Many of them are far from normality.

In the following, we will carry out two tests verifying the hypothesis of multivariate normality (MVN).

• One test is based on the coefficient of multivariate kurtosis $\beta_{2,d}$, introduced by Mardia (see: Mardia, 1980; Mardia, Kent and Bibby, 1979). Let δ_i^2 denote the quantity:

$$\delta_i^2 = (\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}). \tag{1}$$

The sample coefficient of kurtosis is defined as:

$$b_{2,d} = \frac{1}{N} \sum_{i=1}^{N} \delta_i^4. \tag{2}$$

Asymptotically the coefficient $b_{2,d}$ is distributed normally with mean $d(d+1)$ and variance $s_b^2 = 8d(d+2)/N$. Thus

$$u^* = (b_{2,d} - d(d+1))/(8d(d+2)/N)^{1/2} \approx N(0,1). \tag{3}$$

The sample coefficient $b_{2,d}$ of multivariate kurtosis for our d-dimensional data equals $b_{2,13} = 471.83$. Under hypothesis of MVN the expected value is $E(b_{2,13}) = 195$. The value

[1] this was done in the Dept. of Genomics, Institute of Genetics and Microbiology, University of Wrocław, where also the characterization of the spider-plot by $d = 13$ variables was proposed and carried out

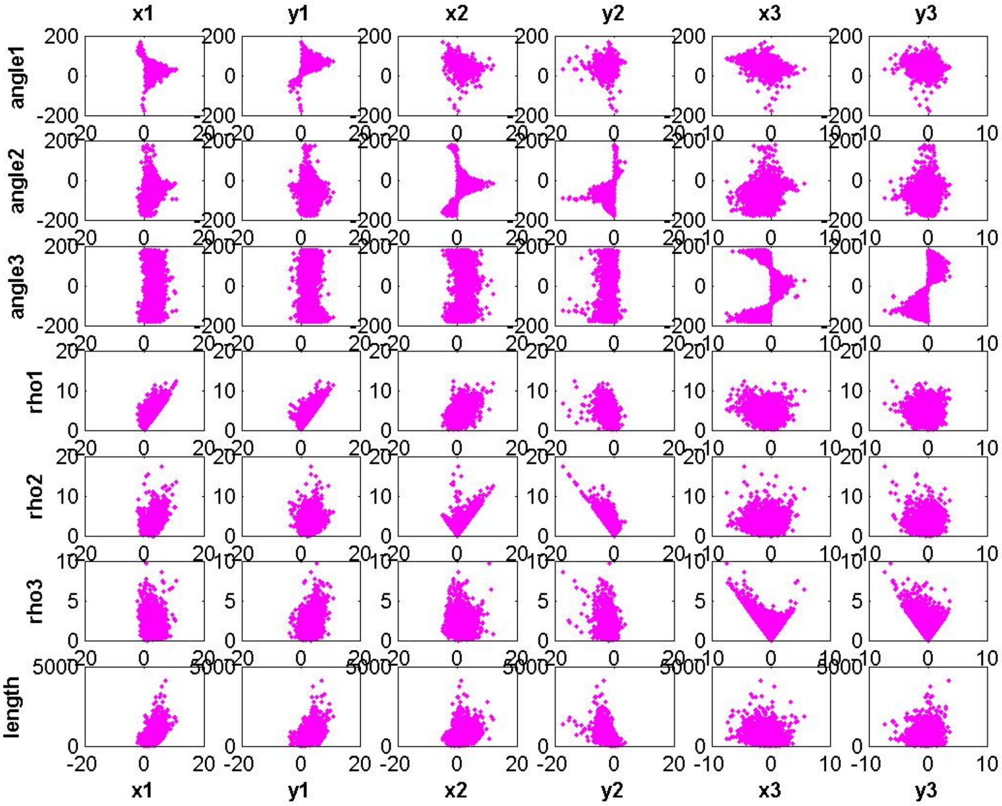


Figure 2: *Scatter-plot matrix displaying pair-wise dependencies among variables* 'y'=angle1, angle2, angle3, rho1, rho2, rho3, length*, put against the variables* 'x' = x1, y1, x2, y2, x3, y3.

of the asymptotic test statistics u^* equals $u^* = 402.63$. Thus the assumption of multivariate normality is decidedly rejected.

• Another test is based directly on the distribution of Mahalanobis distances which were defined by formula (1). Under normality of the sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ the variables δ_i (denoting the so called Mahalanobis distance of the vector $\mathbf{x}_i$ from the center $\bar{\mathbf{x}}$) are – for large N – distributed χ^2_d. The essential point in this approximation is that N should be large enough to give satisfactory approximation of the population mean $\boldsymbol{\mu}$ by the sample mean $\bar{\mathbf{x}}$; similarly, to approximate population covariance matrix $\boldsymbol{\Sigma}$ by the sample covariance matrix $\mathbf{S}$.

N=3300 might be sufficient for this purpose.

Nonetheless, we should be careful, especially when the data set contains some outliers. These might be influential in determining $\bar{\mathbf{x}}$ and $\boldsymbol{\Sigma}$. Therefore it is prudent to use robust estimates when determining the Mahalanobis distances. We have used for that purpose (i.e., for constructing the robust covariance matrix) the *fastmcd* algorithm proposed by Rousseeuw and van Driessen (1999). In Figure 3 we show both robust and ordinary Mahalanobis distances.

Let us remind that both distances – under multivariate normality – should be distributed like the χ^2 statistics with $d = 13$ degrees of freedom. The 0.975 quantile of this distribution equals $\chi^2_{13,.975} = 24.736$. The square root of this quantity equals $4.9735 \approx 5$.

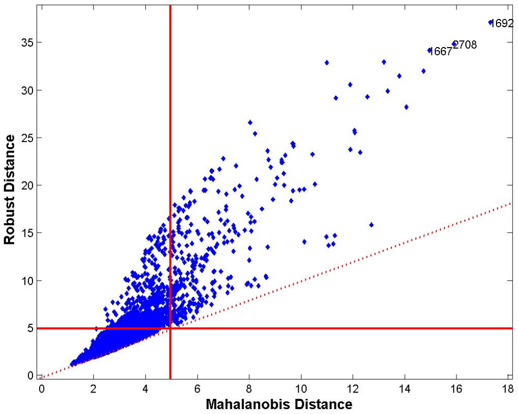


Figure 3. *Robust Mahalanobis distances put against ordinary Mahalanobis distances. One may notice that robust distances are greater then ordinary distances. Robust distances were evaluated using the 'fastmcd' algorithm. The solid lines indicate the square root of the* $\chi^2_{13,.975}$ *quantile.*

The lines $x = 5$ and $y = 5$ are marked in Figure 3 by a solid line. Thus, for a given distance, the expected number of points surpassing that limit should be about 2.5% of the sample size. For our data this amounts about 82.5 data points.

Practically we have 154 robust distances and 155 ordinary distances surpassing the limiting lines $y = 4.97$ and $x = 4.97$. This is decidedly too many. Therefore we should reject the hypothesis that the analyzed data are distributed normally. In fact, they are heavy-tailed.

4 Visualization of the data using Kohonen's self-organizing maps

4.1 The concept of a self-organizing map

Self-organizing maps provide a kind of clustering of the data. The method performs favorably with other methods of clustering. For a comparison, see Dougherty et. al. 2002. The idea of the method is illustrated in Figure 4.

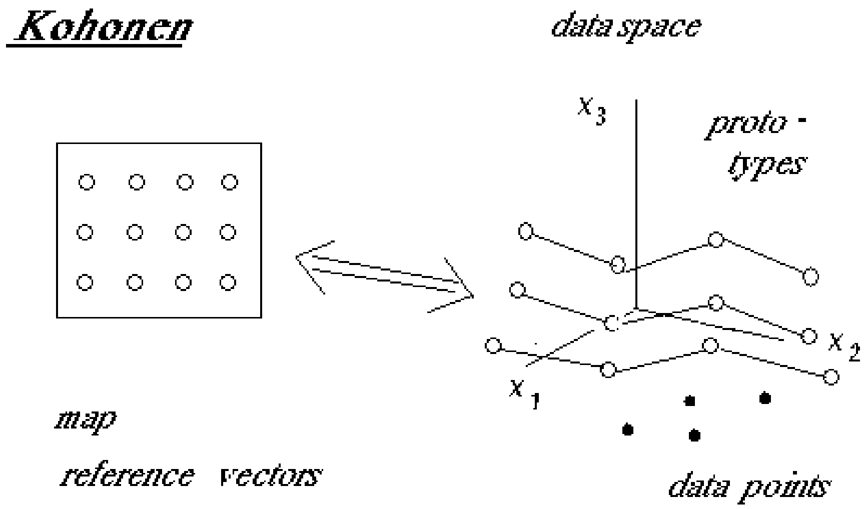


Figure 4. *The idea of a self-organizing map. Points in data space are in one-to-one correspondence with the reference vectors located in the map.*

The map represents usually a planar structure in the form of units arranged as a lattice of size $m1 \times m2$. The nodes of the lattice are characterized by their geometrical positions identified by *reference vectors* $\mathbf{r}_i$. Let us denote the set of the reference vectors by $\mathcal{M} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M)$, with $M = m1 \times m2$ being the total number of the reference vectors, and with $\mathbf{r}_i \in R^2$, $i = 1, \ldots, M$. The reference vectors are ordered by their arrangement in the map. Eg., the map in Figure 4 contains $M = 3 \times 4 = 12$ reference vectors arranged in a rectangular grid (for clarity of exposition, in Figure 4 the grid is invisible). By map units we denote regular (hexagonal or rectangular) areas around the nodes. The map units are numbered similarly as the nodes.

Very important and essential for further considerations is the concept of *neighborhood* of the reference vectors. The neighborhood of a vector $\mathbf{r}_c$ is generally defined by a neighborhood function $h(\mathbf{r}_i, \mathbf{r}_c,)$. The arguments of h are: the central point $\mathbf{r}_c$, ($c \in [1, \ldots, M]$) and the point $\mathbf{r}_i$, for which we want to establish the neighborhood with $\mathbf{r}_c$.

The neighborhood function $h(.,.)$ may be defined in a variety of ways. The property 'belonging to the neighborhood of $\mathbf{r}_c$' may be characterized as a binary property ('yes', 'no') or as a *soft* property described by a real number from the interval [0, 1]. In the last case the function $h(.,.)$ may be viewed as a *membership* function.

We will return to the definition of the neighborhood, when speaking about training the map.

The data space is an Euclidean space R^d with a defined distance between each pair of the data points (data vectors). In our case the data space contains $N = 3300$ points, which represent N data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ from the analyzed data set.

We locate in the data space additionally M prototypes (model vectors), which will be denoted in the following by $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M$ (Kohonen, 1995, calls these prototypes 'codebook vectors'). The M prototypes are in one-to-one correspondence with the reference vectors from the map. The initial position of the prototypes may be quite haphazard, or alternatively, the prototypes may be located in a regular way on the plane spanned by the first two principal components of the data.

The prototypes from the data space are intended to become representatives of the data. To achieve this, they must be trained.

It was stated that the process of training is much faster, when starting from a regular grid of prototypes located on the principal component plane. The obtained results are similar to those obtained when starting from purely random positions, albeit they may not be identical. The procedure of training is heuristic. Using a shrinking neighborhood function and a shrinking learning coefficient (see eq. [4] and [5] below) the process of training yields a stable configuration of the prototypes.

Training the prototypes. During the process of training the prototypes approach the data points and adapt themselves to the (d-dimensional) density distribution of the data. This is done iteratively.

Firstly, for $t = 0$, we set some initial parameters which will be used during the process of training. We initialize the position of the prototypes – giving to them either random values or putting them as ordered points onto the plane spanned by the first two principal components of the data. A distance measure between points in R^d is defined; usually this is the Euclidean distance. Having such measure we are able to assign points of the data space to their nearest representatives. We need also to define the neighborhood function $h(\mathbf{r}_i, \mathbf{r}_j)$. This function depends usually from the iteration step t and reflects the fact that the neighborhood width shrinks with the increase of t. The Matlab SOM Toolbox offers four possibilities: 'gaussian' (default), 'cutgauss', 'ep' (Epanechnikov) and 'bubble'. No obvious superiority of one of these functions over the remaining ones was proved. In our elaboration we have used the Gaussian neighborhood defined by the function

$$h(\mathbf{r}_c, \mathbf{r}_i) = h(\mathbf{r}_c, \mathbf{r}_i, t) = \exp\{-\frac{\|\mathbf{r}_i - \mathbf{r}_c\|^2}{2\sigma(t)^2}\} \tag{4}$$

where $\sigma(t)$ designates the width of the neighborhood function, and is decreasing with t, the iteration step.

Next, for $t = t + 1$ the following procedure is repeated:

set $t = t + 1$; select randomly a data point $\mathbf{x} = \mathbf{x}(t)$; find its closest prototype $\mathbf{w}_c$; update all prototypes $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M$ according to the rule $(i = 1, \ldots, M)$

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) + \eta(t) h(\mathbf{r}_i, \mathbf{r}_c, t)(\mathbf{x}(t) - \mathbf{w}_i(t-1)) \tag{5}$$

where $\eta(t) \in [0, 1]$ is a variable learning rate.

The process of updating is repeated until a STOP criterion is satisfied. The stop criterion may be based on the maximum number of iterations (steps) or on the accuracy of the updated vectors $\mathbf{w}_i(t)$.

The quality of the representation of the data in the map may be characterized by two indices: the quantization error q_e, and the topological error t_e.

The *quantization error* q_e is evaluated as the average distance from each data vector to its nearest prototype (see, e.g., Kohonen, 1995, Vesanto et al., 2000).

The *topological error* t_e is evaluated as the fraction of all data vectors for which the first and second nearest prototype are not represented in adjacent units of the map (see, e.g., Kohonen, 1995, Vesanto, 1999, Vesanto et al., 2000).

The measure t_e was proposed by Kiviluoto (1996). For others concepts of a topological error, see Venna and Kaski, 1999, and the references therein.

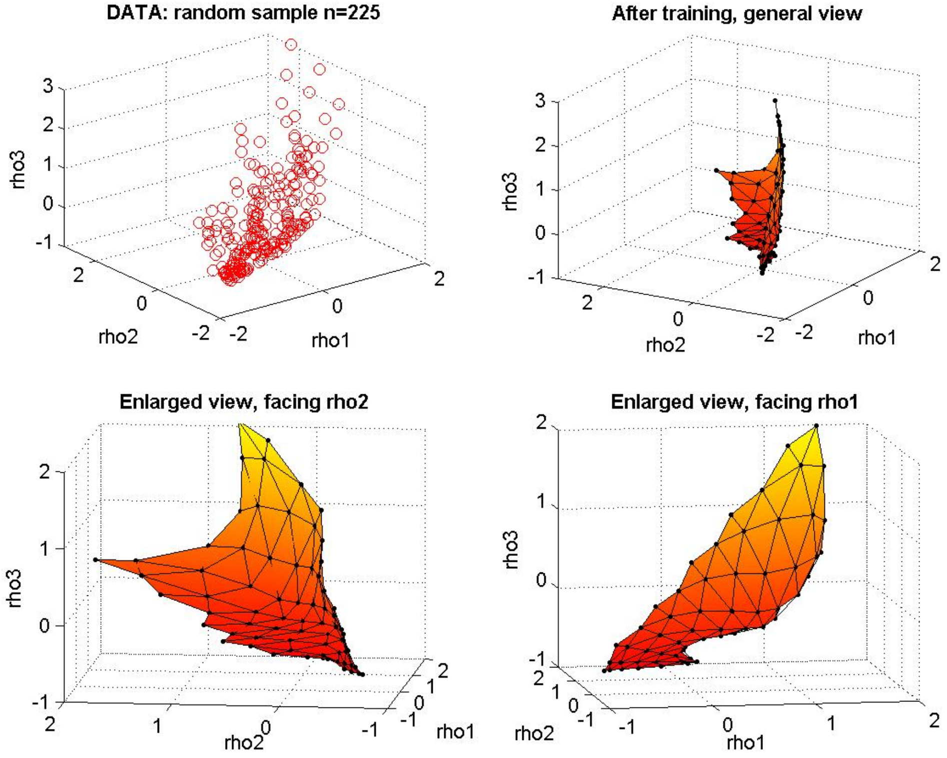


Figure 5. *Training the SOM for 3–dimensional data sampled from the GENES data. Subplot 1 (upper row) shows a 3D plot of the n=225 points included in the sample. Subplot 2 (upper row) shows the prototypes after training – in the same layout, as the data. The prototypes are linked together by segments of various length and designate a complicated surface. Subplots 3 and 4 (bottom row) show enlarged fragments of subplot 2, facing the sides rho2 and rho1.*

In Figure 5 we show an example of constructing the map (SOM) for 3-dimensional data. We took for that purpose a sample of the GENES data containing n=225 data vectors (the GENES data of size 3300 × 13 were first standardized to have column means equal to zero and unit variances). We took for this example only 3 variables: rho1, rho2 and rho3. In calculations we have used the default values of the package Matlab SOM Toolbox (Vesanto et al., 2000). The proposed (default) configuration of the map was: 12 × 6. The starting values of the prototypes were designated by default as regular points located on the PC plane. We have chosen the training option 'long'.

The adaptation of the prototypes to the data was achieved in one instant, although the rough training phase needed 16 and the fine tuning phase 52 cycles, adjusting the location of the prototypes. We have got configuration of the prototypes shown in Figure 5, subplots 2–4. The quality of the representation, expressed by the indices q_e and t_e, is: $q_e = 0.259,\ t_e = 0.040$.

The first exhibit in Figure 5 shows a 3D plot of the data. The second exhibit shows the final configuration of the prototypes. Initially, the prototypes were positioned on a plane and linked together by a regular grid. During the process of training, the prototypes have adapted themselves to the data. With movements of the prototypes, the initial plane has swelled, bulged and recurved to a complicated surface. However, the initial connections ordering the prototypes have remained. The final positions of the prototypes, together with their links, are shown in the second exhibit. Now we see a complicated surface. More detailed views of the enlarged surface – when looking from the rho2 and rho1 axis – are shown in subplot 3 and subplot 4 of the figure.

In Figure 5 we see the true position of the prototypes in the data space. This is possible only for 3-dimensional data, however what we will say now, is true for any dimension, when starting from the PC plane (option 'lininit' in the Matlab SOM Toolbox). Initially, at the start, the prototypes may be imagined to be linked - by stretchable links of equal length, and composing the map (see, e.g., Figure 6a). During the adaptation of the prototypes to the data, the links remain, however they may stretch, as necessary. After the training we stated that the distances between pairs of points have changed, expressing their true distances in the data space. Despite this stretching, the prototypes remained located on a 2–dimensional manifold embedded in the d-dimensional data space. The one-to-one correspondence between each prototype in the data space and each reference vector in the map has remained too.

4.2 Constructing the map for the GENES data

The calculations were done for standardized data using the Matlab SOM Toolbox (see Vesanto et al., 2000) and using its default values. Before starting the analysis, the data were standardized to mean equal zero and variance equal 1, – for each variable. The package proposed for the data a map of size 19 × 15. However, for an easier presentation of the maps, we constructed for our purpose a smaller map of size 13 × 11. This resulted in a deterioration of the quantization error q_e, and in an amelioration of the topological error t_e. Both are shown in the table below.

Table 1. Quality of representation (when starting from the PC plane)

	map size 19×15		map size 13×11	
er-	error		error	
ror	initial	after training	initial	after training
q_e	2.400	1.562	2.408	1.771
t_e	0.000	0.072	0.000	0.066

The grid of the constructed map of size 13 × 11 is shown in Figure 6a. It is a grid based on the hexagonal lattice ('hexa'). Each node of the grid is identified with one reference vector. There is a one-to-one correspondence between the reference vectors in the map and the prototypes in the data space.

During the process of training the prototypes have mixed with the data points. A kind of clustering of the data space has occurred. The data space has been subdivided into M regions (called Voronoi regions), each containing one prototype playing the role of the representative for all data vectors contained in that region.

We may display in the map various information about the data. For instance, for each pair of neighboring map units, we have a corresponding pair of prototypes in the data space. Let us take two neighboring map units. How distant - in the data space - are the two prototypes corresponding to these map units? This can be shown by appropriate shadowing of the map and is shown – for our data – in Figure 6b. Dark hues indicate that the corresponding prototypes are distant, bright hues mean that the prototypes are close. The scale of the distances is shown in the color-bar put at the right of the figure. Looking at the shadowing in Figure 6b, we see there two floating bright regions divided – in the middle of the map – by a dark valley. The southern regions of the map are definitely dark, which indicates for an elongation of the data cloud. The northern regions remain middle light, especially the north-west corner. The north-east corner is dark, which again indicates for growing distance between the prototypes.

It might be also interesting to know how many data vectors are represented by each prototype. This information may be obtained using the technique 'som_hits'. We pass through the data and find for each data vector $\mathbf{x}_i$, $(i = 1, \ldots, N)$ its closest prototype. Say, for $\mathbf{x}_i$ this is the prototype $\mathbf{w}_h$. Then the correspondent reference vector is $\mathbf{w}_h$, and it is contained in map unit no. h. Shortly it can be said that the map unit no. h was hit by the vector $\mathbf{x}_i$. After passing through the entire set of data we obtain the vector $\mathbf{h} = [h_1, \ldots, h_M]$ of hits. The ordering of the hits is the same as that of the reference vectors. The number of hits may also be displayed in the map: either in the form of shadowed hexagons, or in the form of labels expressing digitally the number of hits. The two possibilities are shown in Figure 6c and 6d. One may see that the number of hits varies from 3 to 76.

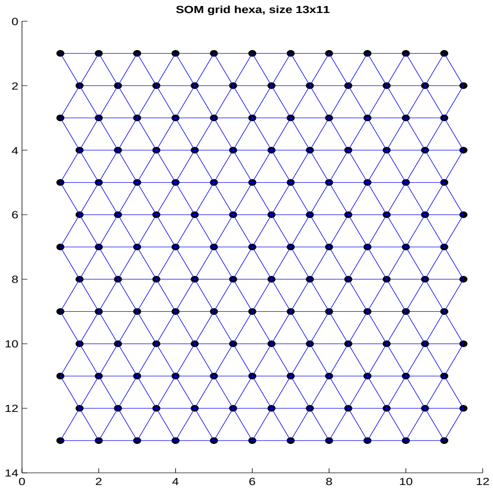


Figure 6a. *The grid (lattice) of a hexagonal map of size* 13×11*. Each node of the grid corresponds to a prototype (codebook vector) located in the data space.*

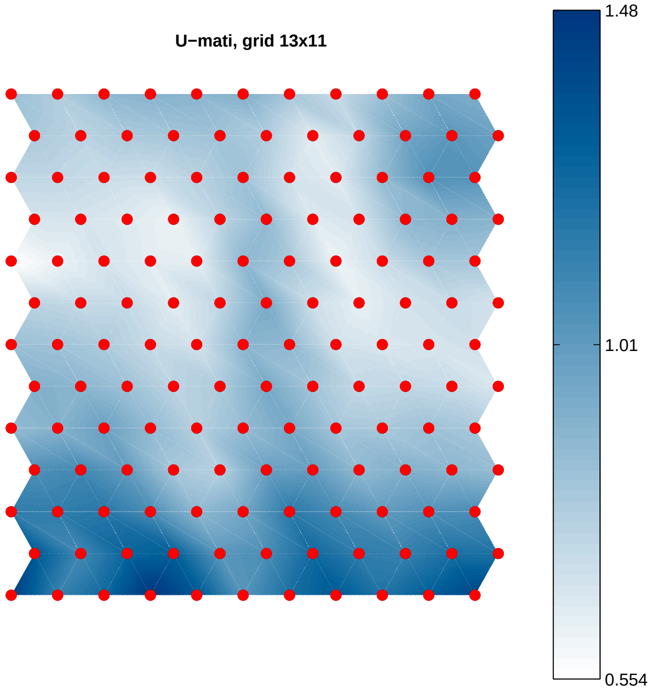


Figure 6b. *Displaying the distances of the prototypes located in the data space. The colorbar located at right indicates the distances. Dark hues mean big distances. The dark dots indicate the reference vectors positioned in the map.*

The numeration of the map units (and also of the components of **h**) is column-wise. For our map of size 13 × 11 the unit in the upper left corner has no. 1; that in the bottom left corner — no 13; those in the right upper and bottom corners got no.s 131 and 143 respectively.

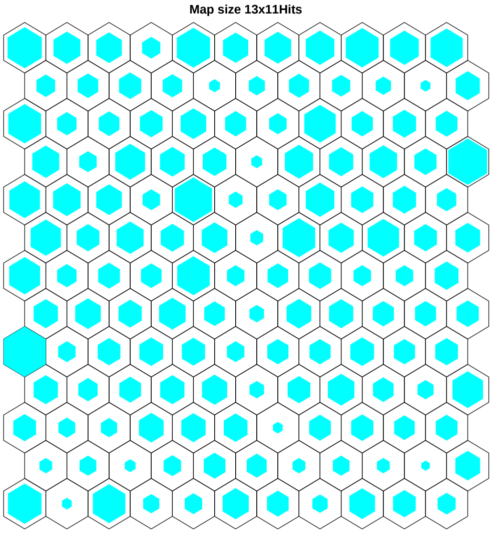


Figure 6c. *Numbers of hits indicated by shadowed sizes of the hexagons.*

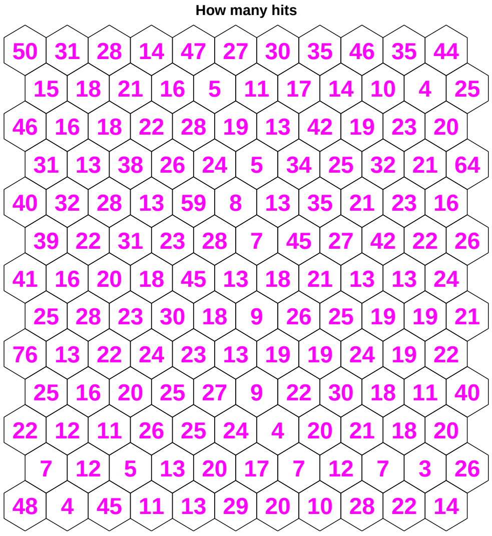


Figure 6d. *Direct displays of the number of hits into map units. The number of hits is differentiated from min=3 to max=76.*

4.3 Which data vectors are represented in corners of the map

The constructed maps reflect the fact that the entire data space was subdivided into M adjacent regions, each region represented by one prototype. How different are the prototypes from various regions?

In the following graph we show prototypes belonging to various corners of the constructed map. Figure 7 exhibits four plots displaying profiles of prototypes of the four corners of the map (remind, our analysis was carried out using standardized data).

In each plot we show three prototypes: that from the true corner, and from two its neighbor units. The values of the neighbors were augmented by 5.0 for one neighbor, and diminished by -5.0 for the other neighbor. In such a way all the three corner prototypes were drawn in one plot.

Looking at the exhibits one may notice that the prototypes shown in the four plots have different patterns, however those belonging to one corner are very similar.

Additionally, for each prototype, we have sampled four data vectors from the region represented by the given prototype. The profiles of these sampled data vectors (appropriately shifted) are also shown in the plots in Figure 8. One may notice that the sampled data vectors exhibit a high degree of similarity to their prototypes.

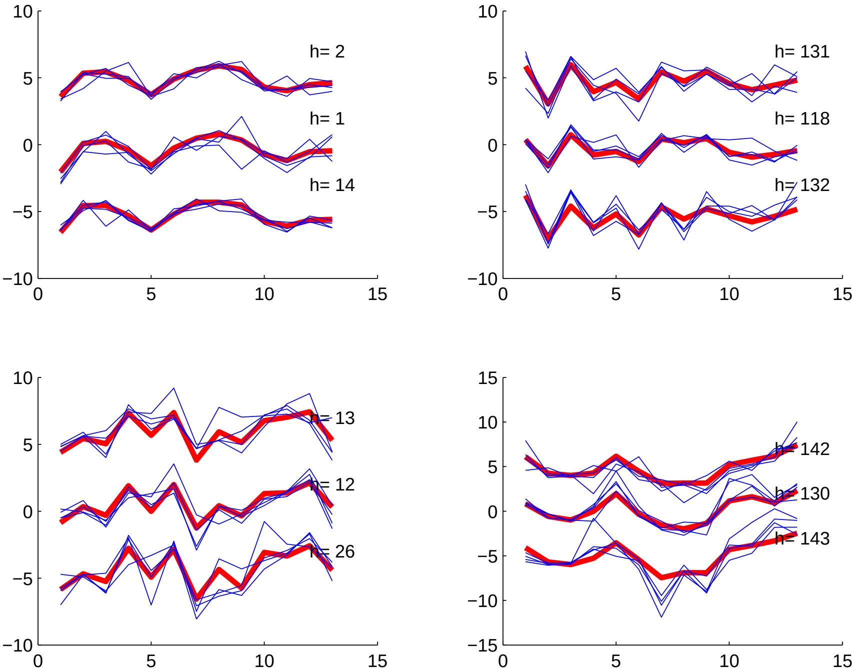


Figure 7. *Profiles of prototypes located in the four corners of the map. The variable* h *denotes the no. of the prototype. From each region – represented by the prototype no. h – four sample vectors were drawn; one may notice that their profiles are similar to those of their prototypes. The x-axis indicates the no. of the variable (V1–V13). The y–axis indicates the values of subsequent variables, as found in the standardized data, eventually augmented by +5 or -5 for neighboring data points (see explanation in text).* {*corner.eps*}

4.4 Interpretation of directions of the map

To construct the map we have used data with 13 variables. One might ask: What is the distribution of values of individual variables over the map.

This might be answered by superposing over the map a color shadowing corresponding to levels of the considered variable.

We have done it and got, a.o., the following results:

1. The primary direction of the map is established by the variable V10 denoting length of the ORF. The highest values appear in the south, and the lowest – in the north of the map.

2. The variable $rho1$ changes systematically from very low values in the north to high values in the south.

3. The relations among $x1, y1$ and $angle1$ are more complicated; none the less, the highest values of $x1$ occur in south-east, and those of $y1$ in south-west.

4. $Angle3$ and $y3$ subdivide the map into two parts: the eastern and the western part, speaking generally.

The corresponding graphs, called *component planes*, one for each variable, are available – in color – at http://ii.uni.wroc.pl/~aba/papers.html.

5 Location of NON-GENES in the SOM established for GENES

In section 4 we have constructed a self-organizing map of size 13 × 11 using the GENES data. The map reflects the fact that the entire data space was subdivided into 13 × 11 adjacent regions, and the entire data set has been subdivided into 13 × 11 clusters located in the data space R^{13}.

Now we will consider the NON-GENES data defined in Section 2. This set, containing n=3030 data vectors, may contain both true genes not recognized in 2000, and ORFs which are actually not genes and will never be recognized as such.

We asked: Is the distribution of the data vectors contained in the NON-GENES set similar to that of the GENES set, elaborated previously?

To answer that question we made an allocation of the NON-GENES data vectors to the regions in R^{13} established for the GENES data. The resulting maps are shown in Figs 8a and 8b.

Looking at the plots one may notice two facts:

1. Generally the NON-GENES data points are spread all over the map.

2. There is an unusually big concentration of points at the northern borders of the map.

The first fact could mean that a great part of the data vectors belonging to the NON-GENES set might denote data items (ORFs), which truly are genes, however in 2000 were not yet recognized as such.

The second fact could mean that a part of the data vectors belonging to the NON-GENES set has actually a different nucleic structure than the GENES data. Assuming that the distribution of known genes is representative for all protein–coding sequences in the yeast genome, we have compared the distribution of data for known genes presented in Figure 6d with the distribution of NON-GENES shown in Figure 8b. We have estimated that in the latter set there are 2081 potential genes, which gives that the total number of protein coding sequences equals 5381. This number is very close to the estimation done by other methods (see Mackiewicz et al., 2002, and the references therein). The data and further information about coding probability for each ORF are available at http://smorfland.microb.uni.wroc.pl/.

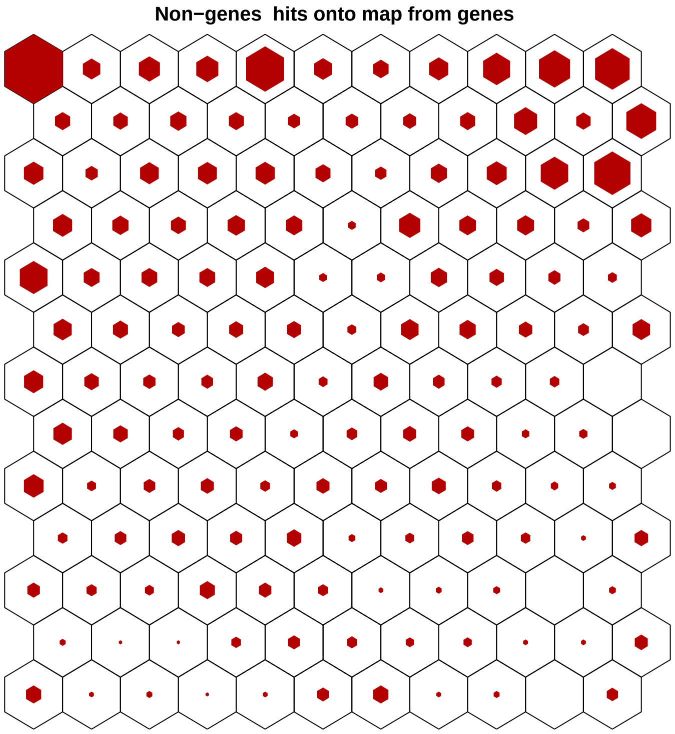
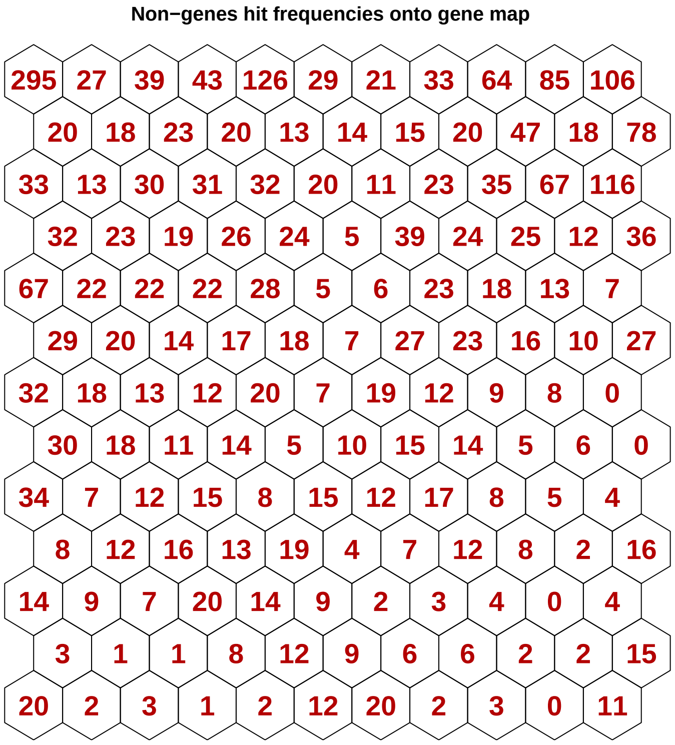


Figure 8a. *Hits of NON-GENES data vectors into units of the map constructed for GENES. The number of hits is indicated by sizes of the hexagons.*



295	27	39	43	126	29	21	33	64	85	106
20	18	23	20	13	14	15	20	47	18	78
33	13	30	31	32	20	11	23	35	67	116
32	23	19	26	24	5	39	24	25	12	36
67	22	22	22	28	5	6	23	18	13	7
29	20	14	17	18	7	27	23	16	10	27
32	18	13	12	20	7	19	12	9	8	0
30	18	11	14	5	10	15	14	5	6	0
34	7	12	15	8	15	12	17	8	5	4
8	12	16	13	19	4	7	12	8	2	16
14	9	7	20	14	9	2	3	4	0	4
3	1	1	8	12	9	6	6	2	2	15
20	2	3	1	2	12	20	2	3	0	11

Fig 8b *Digital displays of the number of hits of the NON-GENES data vectors into units of the map constructed for the GENES data.*

6 Discussion and concluding remarks

We have presented the method of Kohonen's self-organizing maps (SOMs) and shown its performance on a true data set describing 6330 Open Reading Frames (ORFs) of the yeast genome.

We have shown that the SOM method permits for a meaningful graphical visualization of multivariate data cloud in a planar 'map'. In our case data points from R^{13} could be represented in a 2-dimensional plane.

We have shown also that the SOM method enables finding objects which have similar morphology. In the analyzed data we were able finding in an easy way groups of genes, which have similar prevalence of purines and pyramidines in the first and second nucleotide of their codons.

There are many clustering methods, however the SOM method belongs to the few ones which permit for a simultaneous visualization of the clusters with preserving their topology in the multivariate data space.

References

[1] Avery P.J., Henderson D.A. (1999) Fitting Markov chain models to discrete state series such as DNA sequences. *Appl. Statistics*, 48, Part 1, 53–61.

[2] Bartkowiak A., Szustalewicz A., Evelpidou N., Vassilopoulos A. (2003) Choosing data vectors representing a huge data set: a comparison of Kohonen's maps and the neural gas method. Proc. *F*irst Int. Conf. on Environmental Research and Assessment, Bucharest, Romania, pp. 561–572, print on CD–ROM, ©Ars Docendi P. H, Bucharest, Romania.

[3] Braun J.V., Müller H-G. (1998) Statistical methods for DNA sequence segmentation. *Statistical Science*, 13, No. 2, 142–162.

[4] Cebrat S., Mackiewicz P., Dudek M.R. (1998) The role of the genetic code in generating new coding sequences inside existing genes. *Biosystems*, **45** (2), 165–176.

[5] Cebrat S., Dudek M.R. (1998) The effect of DNA phase structure on DNA walks. The European Physical Journal B., **3** 271–276.

[6] Cebrat S., Dudek M., Rogowska A. (1997) Asymmetry in nucleotide composition of sense and antisense strands as a parameter for discriminating open reading frames as protein coding sequences. *J. Appl. Genet.* 38, No. 1, 1–9.

[7] Cebrat S., Dudek M.R., Mackiewicz P., Kowalczuk M., Fita M., (1997) Asymmetry of coding versus noncoding strand in coding sequences of different genomes. *Microbial* & *Comparative Genomics*, 2, No. 4, 259–268.

[8] Dougherty E.R., Barrera J., et al. (2002) Inference from clustering with application to gene-expression microarrays. *Journal of Computational Biology* **9**, Nb. 1, 105–126.

[9] Kamb A., Wang Ch., et al. (1995) Software Trapping: A strategy for finding genes in large genomic regions. *Computers and Biomed. Research* 28, 140–153.

[10] Kiviluoto K. (1996) Topology preservation in self-organizing maps. *Proceedings ICNN'96*, V. 1, June 1996, IEEE Neural Networks Council, Piscataway, New Jersey, USA, 294–299.

[11] Kohonen T. (1995) *Self-Organizing Maps.* Springer Series in Information Sciences, V. 30, Berlin.

[12] Mackiewicz P., Kowalczuk M., Mackiewicz D., Nowicka A., Dudkiewicz M., Laszkiewicz A., Dudek M.R., Cebrat S. (2002) How many protein–coding genes are there in the Saccharomyces cerevisiae genome? *Yeast* **19**(7), 619–629.

[13] Mardia K. V. (1980) Tests of univariate and multivariate normality. In: P. R. Krishnaiah (Ed.), *Handbook of Statistics.* Vol. 1. North-Holland Publishing Company. 279–320.

[14] Mardia, K. V., Kent, J. T., Bibby, J. M. (1979) *Multivariate Analysis.* Academic Press, London.

[15] Muri, F. (1998) Modelling bacterial genomes using hidden Markov models. *C*OMPSTAT 1998, Invited and Contributed Papers, 89–100.

[16] Prum B., Rodolphe F., de Turckheim E. (1995) Finding words with unexpected frequencies in desoxyribonucleic acid sequences. *J.R. Statist. Soc. B*, 57, No. 1., 205–220.

[17] Rousseeuw P.J., van Driessen K. (1999) A fast algorithm for the minimum covariance determinant. *Technometrics* **41**, pp. 212–223.

[18] Venna J., Kaski S. (2001) Neighborhood preservation in nonlinear projection methods: An experimental study. In: G. Dorffner, H. Bischof, and K. Hornik (Eds), *Proceedings ICANN 2001*, Vienna. Springer, Berlin, pp. 485–491.

[19] Vesanto J. (1999) SOM–based data visualizing methods. *Intelligent Data Analysis*, **3**(2), pp. 111–126.

[20] Vesanto J., Himberg J., Alhoniemi E., Parhankangas J. (2000) *SOM Toolbox for Matlab 5.* SOM Toolbox Team, Helsinki University of Technology, Finland, Libella Oy, Espoo, pp. 1–54. `http://www.cis.hut.fi/projects/somtoolbox`, Version 0beta 2.0, Nov. 2001.