Landscape Multidimensional Scaling

Katharina Tschumitschew[1], Frank Klawonn[1], Frank Höppner[2], and Vitaliy Kolodyazhniy[3]

[1] Department of Computer Science
University of Applied Sciences Braunschweig/Wolfenbüttel
Salzdahlumer Str. 46/48
D-38302 Wolfenbuettel, Germany
{katharina.tschumitschew,f.klawonn}@fh-wolfenbuettel.de

[2] Department of Economics
University of Applied Sciences Braunschweig/Wolfenbüttel
Robert Koch Platz 10-14
D-38440 Wolfsburg, Germany
f.hoeppner@fh-wolfenbuettel.de

[3] Institute for Psychology
University of Basel
Missionsstrasse 60/62
Basel, CH-4055, Switzerland
v.kolodyazhniy@unibas.ch

Abstract. We revisit the problem of representing a high-dimensional data set by a distance-preserving projection onto a two-dimensional plane. This problem is solved by well-known techniques, such as multidimensional scaling. There, the data is projected onto a *flat plane* and the Euclidean metric is used for distance calculation. In real topographic maps, however, travel distance (or time) is not determined by (Euclidean) distance alone, but also influenced by map features such as mountains or lakes. We investigate how to utilize *landscape features* for a distance-preserving projection. A first approach with rectangular cylindrical mountains in the MDS landscape is presented.

Key words: visualisation, multidimensional scaling, landscape multidimensional scaling

1 Introduction

Large data sets call for tools that (semi-) automate as many of the tedious steps in data analysis as possible, but usually cannot replace the visual inspection of data or results, because the visual perception of humans is extremely good in detecting abnormalities that are difficult to detect automatically. This explains why visualisation techniques are useful and important, even in times of powerful data mining techniques and large data sets.

In this paper, we therefore revisit the problem of mapping high-dimensional data to a two- or three-dimensional representation. There are various approaches

to solve this problem: a data record may be represented by a pictogram, where each attribute is mapped to a detail of the pictogram (e.g. stick figures or Chernoff faces), a record may be represented by a sequence of line segments (e.g. one line segment per attribute as with parallel coordinates), or it may be projected into a low-dimensional space and then represented by a dot in a scatter plot. In the latter category the main objective is usually to preserve either the variance of the data (principal component analysis (PCA) [4]) or the distances between the data objects (e.g. multidimensional scaling (MDS) [7], and modifications thereof [8]). For most of these techniques the graph consists of one graphical element per data record (pictogram, sequence of lines, dot). Only with very few visualisation techniques additional elements provide further information about the depicted data. Just like in a map with level curves, where the existence of mountains between two geographical positions indicate a longer traveltime, we want to understand the graph as a landscape that carries information in its own right.

The paper is organized as follows: In section 2 we discuss the kind of landscape we will consider and justify it by some theoretical considerations. We stick to the idea of a distance-preserving map and, starting from a flat map (discussed in section 3), we incrementally modify the landscape to improve the overall error (section 4). Some results and examples are provided in section 5.

2 MDS Representation on a Landscape

We assume that a p-dimensional data set $X = \{x_1, x_2, ..., x_n\} \subseteq \mathbb{R}^p$ is given. By $d(v, w)$ we denote the Euclidean distance $\|v - w\|$. In MDS each of the high-dimensional data points x_i has to be mapped to a low-dimensional representative y_i. The projection of X is denoted as $Y = \{y_1, y_2, ..., y_n\} \subseteq \mathbb{R}^q$ where $1 \leq q < p$ (typically $q \in \{2, 3\}$). A perfect distance-preserving projection of X to Y would keep the distances $d_{ij}^x := d(x_i, x_j)$ of the high-dimensional space identical to the distances $d_{ij}^y := d(y_i, y_j)$ of the projected data objects, that is, $d_{ij}^x = d_{ij}^y$ holds. A perfect projection is, however, impossible except for a few trivial cases. Therefore, MDS seeks to minimize the error introduced by the projection ($|d_{ij}^x - d_{ij}^y|$ for all i, j). Common objective functions are [7]:

$$E_1 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(d_{ij}^x\right)^2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(d_{ij}^y - d_{ij}^x\right)^2, \tag{1}$$

$$E_2 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_{ij}^y - d_{ij}^x}{d_{ij}^x} \right)^2, \tag{2}$$

$$E_3 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}^x} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{\left(d_{ij}^y - d_{ij}^x\right)^2}{d_{ij}^x}. \tag{3}$$

Usually the selected *stress function* is minimized by a numerical optimisation method such as gradient descent.

A similar technique that does not use any of the above objective functions is the self-organizing map (SOM) [6]. The data objects are assigned to cells on a regular grid such that data objects in neighbouring cells are similar to each other. Traditionally, the colouring of the cells is used to provide additional information about the similarity to data in adjacent cells. In [9] an extension has been proposed that encodes information about the data density as well as the distance between cell data in a *landscape*, the so-called U^*-matrix. The distance of neighbouring cells is reflected by the landscape, but for non-adjacent cells no conclusions can be made.

In this work we stick to a distance-preserving map (just as with MDS), but we want to use the landscape as an additional parameter influencing the *perceived distance* between data objects placed in the landscape. With a real map the true travel distance depends on the chosen path: we may circumvent or climb a mountain, for instance. To be of immediate use no tedious *path optimisation* should be necessary to understand the visualisation, therefore only the straight connection between the points is considered relevant for their distance. If the straight line between two projected points is shorter than the distance between their high-dimensional originals, we may introduce obstacles on the path to increase their map-distance. From the number and height of the obstacles a user gets a better impression of the true distance.

A landscape MDS (LMDS) may be constructed in the following way: We seek for an initial distribution of data objects on a flat plane guaranteeing that

$$d_{ij}^y \leq d_{ij}^x \tag{4}$$

holds for all $i, j \in \{1, \ldots, n\}$. This condition is motivated by the fact that mountains can only increase the distances. Then we place rectangular or cylindrical mountains (parameterized by location, size and height) in the landscape such that the difference $|d_{ij}^y - d_{ij}^x|$ is reduced. In our simple first model a mountain increases the path length by twice the height of the cylindrical mountain, corresponding to climbing the mountain and descending to the base level again.

Can such a *landscape MDS* deliver better results than traditional MDS? Yes, it can, at least in theory. We place the projections y_i on a circle such that the distances are no larger than the original ones and no three points are collinear (figure 1, left). We ensure $d_{ij}^y \leq d_{ij}^x$ by making the circle sufficiently small. Then introduce cylindrical mountains m_{ij} $(i < j)$ such that each mountain m_{ij} is crossed only by the line connecting the two points y_i and y_j (figure 1, right). Choosing $h_{ij} = (d_{ij}^x - d_{ij}^y)/2$ reduces the error of the distances to zero in the landscape.

The drawback of this solution is obviously that in the worst case $n(n-1)/2$ very narrow cylindrical mountains for n data objects have to be introduced. The resulting landscape is very different from common maps and thus hard to interpret. We will develop a better alternative in the following sections.

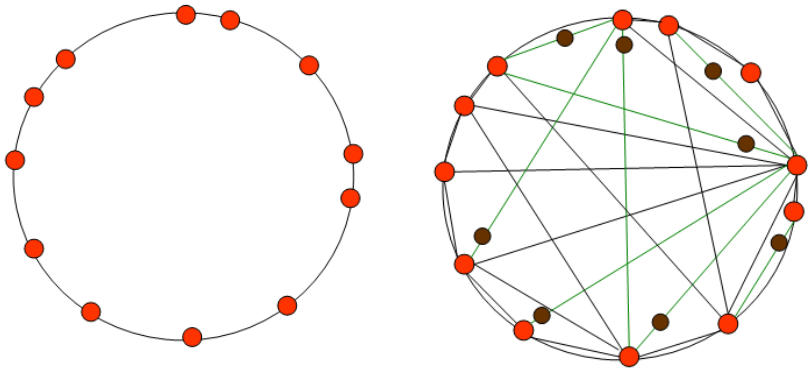

Fig. 1. An optimal solution with zero error.

3 Initialisation of LMDS

As mentioned above, landscape MDS (LMDS) must be initialized such that the constraints (4) for the distances are fulfilled. There are a number of approaches which can be used for the initialization of an LMDS map. To choose the best initialization strategy, we performed a comparison of the following four approaches:

(a) Projecting the data to the plane by PCA approach automatically satisfies the constraint (4) of smaller distances.
(b) The second approach enhances the first one by shifting points within allowed intervals after the PCA projecting. The intervals are computed such that the error function (3) is minimized and the violation of the constraints (4) is avoided. The interval computation involves the golden section search algorithm.
(c) Classical MDS which violates the constraint of smaller distances.
(d) MDS with constraints on distances using a Lagrange function which is defined as

$$L = E_3 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \lambda_{ij} \left(d_{ij}^y - d_{ij}^x \right). \tag{5}$$

Although no analytical solution of (5) exists, the Lagrange function can be optimized iteratively using the Arrow-Hurwitz-Uzawa algorithm [2]:

$$\begin{cases} \lambda_{ij}^{(k+1)} & = \max\left\{0,\ \lambda^{(k)} + \alpha^{(k)} \left(d_{ij}^y - d_{ij}^x \right) \right\}, \quad i,j = 1,\ldots,n \\ y_i^{(k+1)} & = y_i^{(k)} - \beta^{(k)} \frac{\partial L\left(y^{(k)},\, \lambda^{(k)}\right)}{\partial y_i}, \qquad\qquad i = 1,\ldots,n \\ 0 < \alpha^{(k)} \leq 1 & \\ 0 < \beta^{(k)} \leq 1 & \end{cases} \tag{6}$$

where $\alpha^{(k)}$ and $\beta^{(k)}$ are the step length parameters. The undetermined Lagrange multipliers $\lambda_{ij}^{(k+1)}$ are equal to zero when the constraints (4) are satisfied, and become positive when the constraints are violated.

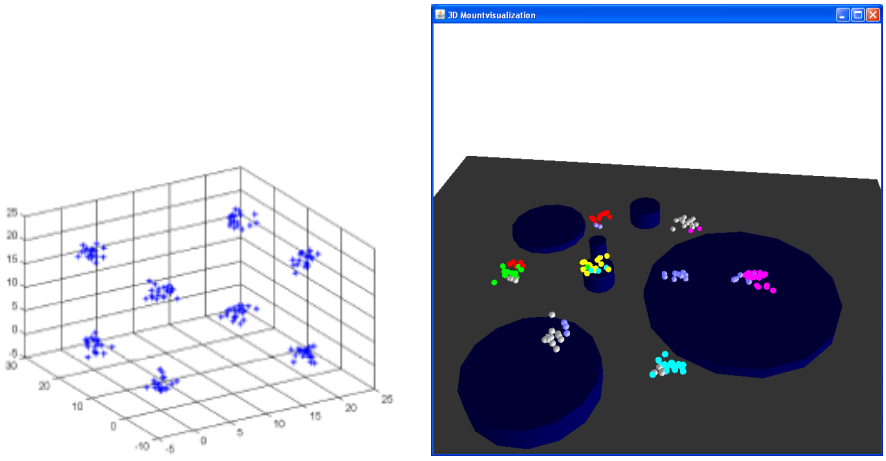

Fig. 2. The cube data set.

To compare the initialization results provided by the different approaches listed above, we used a number of benchmark data sets: 'Iris' (4 attributes) and 'Glass' (9 attributes) from the UCI Machine Learning Repository [1] with 150 and 214 data objects, respectively, as well as the two three-dimensional data sets 'Cube' and 'Coil' (see the left images in figures 2 and 3, respectively.) with 360 and 160 data objects, respectively. The results for these data sets provided by the considered initialization methods are listed in table 1. The number of iterations required for PCA with shifting was 5, because more iterations did not significantly improve the results. For constrained MDS with the Arrow-Hurwitz-Uzawa algorithm, the number of iterations was 5000.

As can be seen from table 1, constrained MDS with the Arrow-Hurwitz-Uzawa optimisation procedure considerably outperforms the other three approaches w.r.t. the objective function (3) and avoiding the violation of constraints (4). So we will use this technique for the initialization of LMDS in the following section. The important advantage of the procedure (6) are its high speed of convergence and its low computational costs.

4 Adding Mountains to the Landscape

In the previous section, initialisation strategies were compared that position the data points on a flat landscape in such a way that the constraints (4) are

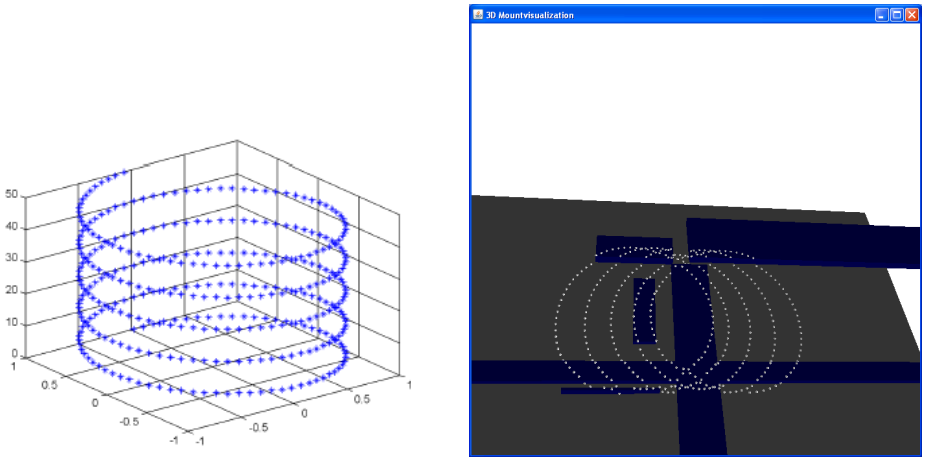

Fig. 3. The coil data set.

(almost) satisfied. Therefore, in the next step, mountains will be introduced that can only lead to in an increase of the distances of the flat landscape. The effect of a cylindrical or rectangular mountain of height h on the distance between two points i and j is

$$d_{ij}^{new} = d_{ij} + \xi_{ij} h, \qquad \xi_{ij} \in \{0,\, 1,\, 2\}\,. \tag{7}$$

Figure 4 shows the possible cases and the values for ξ_{ij}.

When we add mountains to the landscape, we have to decide where to place the mountains (the underlying rectangle for a rectangular mountain and centre point as well as the radius for a cylindrical mountain), determine their heights and also the number of mountains we want to introduce. Again, there is no analytical solution to this problem. The objective function is not even continuous

Table 1. Empirical comparison of the initialisation strategies.

Data set	**PCA:** **Error (3)**	**MDS:** **Error (3)** **Sum of violations of constraint (4)**	**PCA with shifting:** **Error (3)** **Sum of violations of constraint (4)**	**Arrow-Hurwitz-Uzawa MDS:** **Error (3)** **Sum of violations of constraint (4)**
Iris	0.0097589 0	0.00632271946452 453.0641	0.009536704494 0	0.0090821097634 3.085490787E-4
Coil	0.0768078 0	0.0546562810427 8805.257	0.076738333 0	0.0746009358153 8.20138002E-4
Glass	0.170403 0	0.03577761897878 4176.65	0.10563177615 0	0.080108479955 7.677778033E-4
Cube	0.070009 0	0.0479180387553 1669.474	0.067812491266 0	0.0652975311919 7.673804646E-4

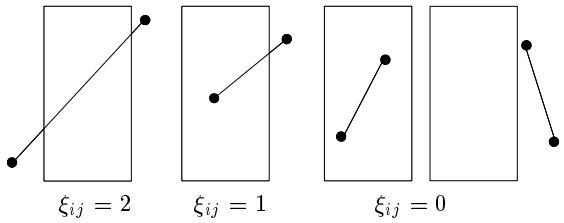

Fig. 4. Effects of a rectangular mountain on the distance between two points.

according the coefficients ξ_{ij}. Therefore, we apply an evolution strategy (see for instance [3]) for positioning the mountains. The height of a mountain is not considered as a parameter of the evolution strategy, since the optimal height for a mountain with a fixed location can be obtained from $\frac{\partial E_3}{\partial h} = 0$, leading to

$$h \;=\; -\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{d_{ij}^{y}-d_{ij}^{x}}{d_{ij}^{x}}\xi_{ij}}{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\xi_{ij}}{d_{ij}^{x}}\xi_{ij}}. \tag{8}$$

There are two strategies to add mountains to the landscape. Mountains can be introduced to the landscape one by one or a fixed number of mountains is added simultaneously. In both cases, the intersection of mountains must be avoided. Otherwise, (7) would not be valid anymore and the computing of the revised distances would become quite complicated. It is, however, allowed that one mountain is placed completely on top of another, as long as there is not just a partial overlap of mountains. When k moutains are added simultaneously and not one by one to the landscape, the optimal heights of the mountains can no longer be computed from (8). The heights are given by the solution of the following system of linear equations.

$$\sum_{l=1}^{k} h_l \sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{\xi_{ijl}}{d_{ij}^{x}}\xi_{ijp} \;=\; -\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{d_{ij}^{y}-d_{ij}^{x}}{d_{ij}^{x}}\xi_{ijp} \qquad (p = 1,\ldots,k) \tag{9}$$

The solution of this system of linear equations can lead to negative heights. Negative heights as well as the avoidance of overlapping mountains is enforced by a dynamic penalty function to the evolution strategy that assigns a low fitness to solutions with intersecting mountains. The penalty function for cylindrical mountains is described here. The corresponding penalty function for rectangular mountains can be defined in a similar way. Two cylindrical mountains with radius r_i and r_j overlap when the distance $\delta^{(ij)}$ between the midpoints of their circles satisfies

$$\delta^{(ij)} \;\leq\; r^i + r^j. \tag{10}$$

If in this case either $\delta^{(ij)} + r^i \leq r^j$ or $\delta^{(ij)} + r^j \leq r^i$ holds, then one of the mountains lies completely on top of the other and no penalty is required. In case

a penalty is needed, it is defined as $g_{ij} = r^i + r^j - \delta^{(ij)}$, otherwise $g_{ij} = 0$ is chosen. The overall penalty is given by

$$\text{pen} = \sum_{i=1}^{k} -\min\{h_i, 0\} + \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} g_{ij}. \tag{11}$$

The penalty function is increased with the number of generations of the evolution strategy, so that in the end solutions with intersecting mountains have no chance to survive due to the bad fitness value. The overall fitness is given equation (3) plus

$$f(t) \cdot \text{pen} \tag{12}$$

where $f(t)$ is a positive and increasing function. t refers to the generation number of the actual population of the evolution strategy.

In order to speed up the computations, the determination of the ξ_{ij} values for rectangular mountains is carried out based on the Cohen-Sutherland line-clipping algorithm (see for instance [5]) used in computer graphics. Line clipping refers to the problem of finding out whether a line has to be drawn in a rectangular window on the computer screen, so that it is necessary to determine, whether the line is completely inside, completely outside or partly inside the window. This corresponds exactly to the problem of finding out whether a connecting line between two points is affected by a rectangular mountain, as it is illustrated in figure 4.

For cylindrical mountains the same strategy is used by applying the Cohen-Sutherland line-clipping algorithm to the bounding square of the circle associated with the cylindrical mountain. In case a line intersects the bounding square, it is still necessary to test whether it intersects also the circle. But at least for lines outside the bounding square, we know $\xi_{ij} = 0$.

Introducing mountains in a greedy manner step by step with the evolution strategy turned out to be faster than adding mountains simultaneously. However, the results of the latter strategy were slightly better in our experiments.

5 Examples

In this section we present the results of numerical simulations of the proposed LMDS visualisation method. We used five data sets: the first four are the artificial data sets 'Cube', 'Pyramid', 'Coil' and 'Ring'. Two of them were already mentioned in section 3. The third data set is a real-world data set from a wastewater treatment (WWT) plant in Germany. The results are shown in figures 2, 3, 5, 6 and 7, respectively. Note that the 3D-effect is better, when the images can be rotated or zoomed. The four artificial data sets are all three-dimensional. On the left-hand side of each figure, the original data set is shown, on the right-hand side the LMDS result.

For the WWT plant, measurements of 15 process variables over a period of six years were available. The main task was to visualize the year-to-year variations

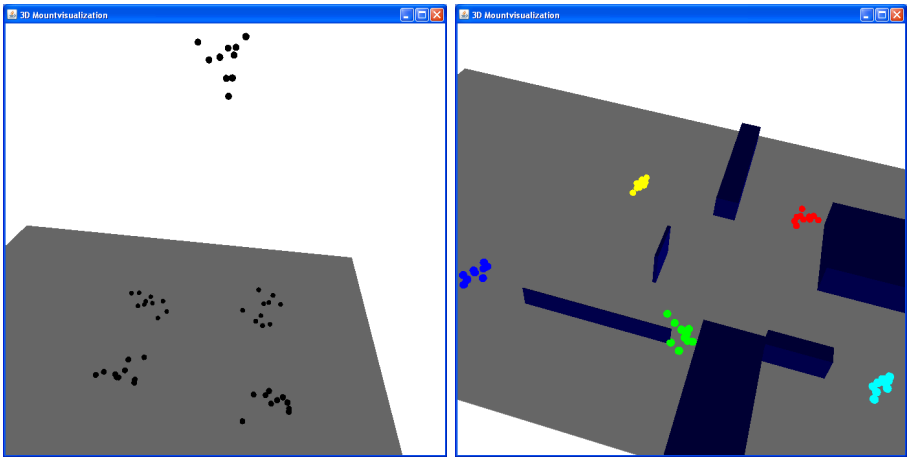

Fig. 5. The pyramid data set.

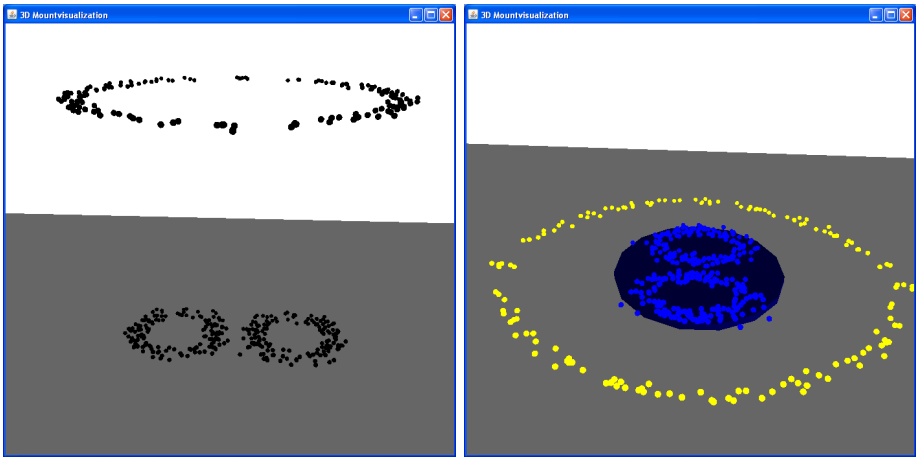

Fig. 6. The ring data set.

in the plant. We did not use any information on time or date for the generation of the visualisation.

The 3D-diagram generated from our method shows a clear separation of the year 1996 which is marked by bigger yellow spheres. This is confirmed by our knowledge of changes in the process that were implemented in the year of 1997.

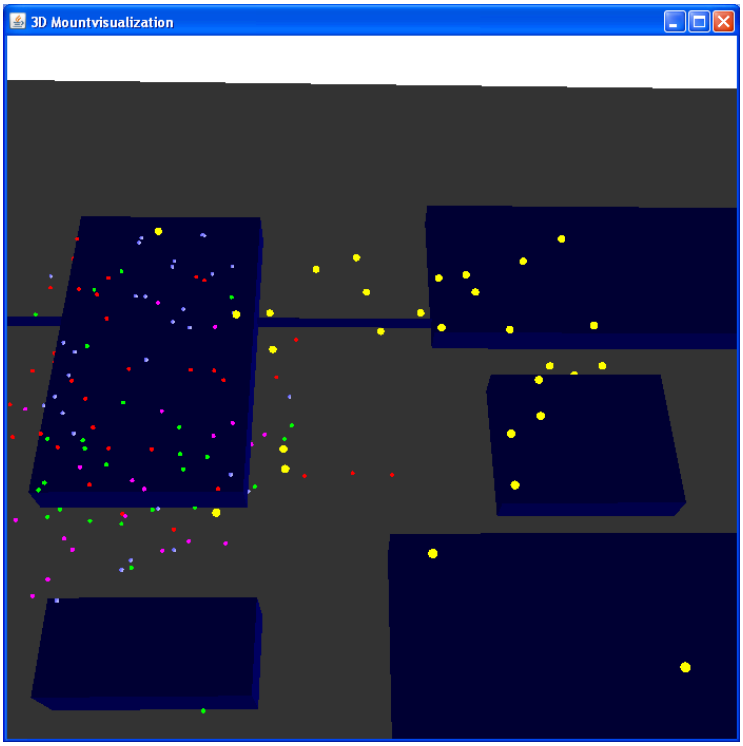

Fig. 7. The wastewater treatment plant data set.

Table 2 shows a comparison of the error values for classical MDS and for LMDS for the objective function (3). The error value of LMDS is not always better. The main reason is the greedy strategy to start with an MDS initialisation with the constraints described in equation (4). It seems that the restricted MDS gets stuck in a local minimum easier and the introduction of mountains afterwards cannot always compensate this effect completely.

6 Conclusions

We have introduced a new method that exploits a landscape for multidimensional scaling instead of a flat plane. In principle, it is possible to position the

Table 2. Comparison of MDS to LMDS

Error according to equation (3)		
Data set	**MDS**	**LMDS**
Pyramid	0.02028127	0.00872146
Ring	0.02551469	0.02430803
Cube	0.04791804	0.03802418
Coil	0.05465628	0.05809677
Wastewater treatment plant	0.07254737	0.08211769

data points on the landscape in such a way that the distances in the original high-dimensional data space are preserved exactly. However, this will lead to extremely complicated and non-intuitive landscapes.

Simplified landscapes allowing only a strictly limited number of mountains suitable for real visualisation purposes cannot guarantee this exact representation of the distances anymore. However, shortening distances instead of increasing them by mountains might lead to better solutions. The reason for this problem can be seen in figure 8. The typical way to fit a two-dimensional plane into a higher dimensional space would be to fold it like a towel, which would lead to larger distances instead of smaller distances as illustrated in figure 8. But mountains in LMDS can only increase, but not decrease distances and concepts like 'wormholes' that would be needed to shorten distances are not very suitable for visualisation purposes.

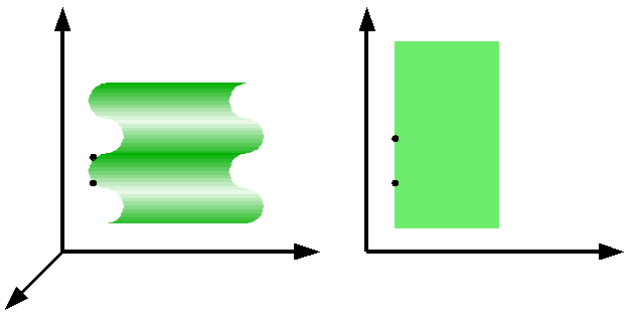

Fig. 8. The problem of fitting a lower into a higher dimensional space.

References

1. UCI machine learning repository. www.ics.uci.edu/~mlearn/MLRepository.html.
2. K. Arrow, L. Hurwitz, and H. Uzawa. *Studies in Nonlinear Programming.* Stanford University Press, Stanford, 1958.

3. T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, Oxford, 1996.
4. G. Dunn and B. Everitt. *An Introduction to Mathematical Taxonomy.* Cambridge University Press, Cambridge, MA, 1982.
5. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice. Second Edition in C.* Addison-Wesley, Boston, 1996.
6. T. Kohonen. *Self organizing maps.* Springer, Heidelberg, New York, 2001.
7. J.B. Kruskal and M. Wish. *Multidimensional Scaling.* SAGE Publications, Beverly Hills, 1978.
8. F. Rehm, F. Klawonn, and R. Kruse. MDS_{polar}: A new approach for dimension reduction to visualize high dimensional data. In A.F. Famili, J.N. Kook, J.M. Peña, and A. Siebes, editors, *Advances in Intelligent Data Analysis V*, pages 316–327, Berlin, 2005. Springer.
9. A. Ultsch. Clustering with SOM: U*C. In *Workshop on Self-Organizing Maps (WSOM 2005)*, pages 75–82, Paris, 2005.