POLARMAP - Efficent Visualization of High Dimensional Data

Frank Rehm, Frank Klawonn, and Rudolf Kruse

***Abstract*— Mulidimensional scaling provides low-dimensional visualization of high-dimensional feature vectors. This is a very important step in data preprocessing because it helps the user to appraise which methods to use for further data analysis. But a well known problem with conventional MDS is the quadratic need of space and time. Beside this, a transformation of MDS must be completely recomputed if additional feature vectors have to be considered. The POLARMAP algorithm, presented in this paper, learns a function, similar to NeuroScale, but with lower computational costs, that maps high-dimensional feature vectors to a 2-dimensional feature space. With the obtained function even new feature vectors can be mapped to the target space.**

I. Introduction

Visualization is an important step as a part of data preprocessing, since the amount of data and their dimensionality is growing fast. A visual assessment of a high dimensional feature space is not possible without the help of appropriate methods. Many such techniques have been published so far. Some of these methods provide dimension reduction, the use of icons or rearrangement of the dimensions. Principal component analysis (PCA) is a very efficient method for dimension reduction that unfortunately is not appropriate for manifolds. Multidimensional scaling (MDS) [1], [5] is very powerful, mapping data to a low-dimensional feature space. Many modifications of MDS are published so far, but high computational costs prevent their application to large datasets [3], [9]. In recent years some research has been done in this regard [2], [7], [10].

In this paper we present a new approach for dimension reduction. Our approach is a modification of published MDS$_{polar}$. Instead of trying to preserve the distances between feature vectors as for MDS, our algorithm transforms high-dimensional feature vectors into 2-dimensional feature vectors under the constraint that the length of each vector is preserved and the angles between vectors approximate the corresponding angles in the original space as good as possible. As an improvement of MDS$_{polar}$, we will present an algorithm that learns a function, that enables the user to map even new feature vectors to the target space. Finally, we will describe a technique to learn such mappings from data with $O(n \cdot \log n)$ space- and time-complexity.

This paper is organised as follows: in section II, we recall the conventional MDS method. In section III, we will briefly describe MDS$_{polar}$, which is the base algorithm for our method, that we will present in section IV. In section V we will discuss some experimental results. Finally, we conclude with section VI.

II. Multidimensional Scaling

Multidimensional scaling (MDS) is a method that estimates the coordinates of a set of objects $Y = \{y_1, \ldots, y_n\}$ in a feature space of specified (low) dimensionality that come from data $X = \{x_1, \ldots, x_n\} \subset \Re^p$ trying to preserve the distances between pairs of objects. Different ways of computing distances and various functions relating the distances to the actual data are commonly used. These distances are usually stored in a distance matrix

$$D^x = \left(d^x_{ij}\right), \quad d^x_{ij} = \|x_i - x_j\|, \quad i,j = 1,\ldots,n.$$

The estimation of the coordinates will be carried out under the constraint, that the error between the distance matrix D^x of the dataset and the distance matrix $D^y = \left(d^y_{ij}\right)$, $d^y_{ij} = \|y_i - y_j\|$, $i,j = 1,\ldots,n$ of the corresponding transformed dataset will be minimised.

Thus, different error measures to be minimised were proposed, i.e. the absolute error, the relative error or a combination of both. A commonly used error measure, the so-called *Sammon's mapping*

$$E = \frac{1}{\sum_{i=1}^{n}\sum_{j=i+1}^{n} d^x_{ij}} \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{\left(d^y_{ij} - d^x_{ij}\right)^2}{d^x_{ij}}$$

describes the absolute and the relative quadratic error. To determine the transformed dataset Y by means of minimising error E a gradient descent method can be used. By means of this iterative method, the parameters y_k to be optimised, will be updated during each step proportional to the gradient of the error function E. Calculating the gradient of the error function leads to

$$\frac{\partial E}{\partial y_k} = \frac{2}{\sum_{i=1}^{n}\sum_{j=i+1}^{n} d^x_{ij}} \sum_{j \neq k} \frac{d^y_{kj} - d^x_{kj}}{d^x_{kj}} \frac{y_k - y_j}{d^y_{kj}}.$$

After random initialization for each projected feature vector y_k a gradient descent is carried out and the distances d^y_{ij} as well as the gradients $\frac{\partial d^y_{ij}}{\partial y_k}$ will be recalculated again. The algorithm terminates when E becomes smaller than a certain threshold.

The complexity of MDS is $O(c \cdot n^2)$, where c is the (unknown) number of iterations needed for convergence of the gradient descent scheme. Thus, MDS is usually not applicable to larger datasets. Another problem of MDS is that it does not construct an explicit mapping from the high dimensional space to the lower dimensional space, but just tries to position the lower dimensional feature vectors in a suitable way. Therefore, when new data have to be considered, they cannot be mapped directly to the lower dimensional space, but the whole MDS

procedure has to be repeated. NeuroScale [6] is a scheme that tries to construct an explicit mapping for MDS in the form of a neural network. However, it does not reduce the complexity of MDS. In [3] a more efficient, but still iterative approach was proposed making use of a step-by-step reduction by one dimension based on determining the best projection in each step.

In [8], a different algorithm is proposed, not needing any iterative scheme and whose complexity can be reduced to $O(n \cdot \log n)$.

III. MDS$_{polar}$ - Multidimensional Scaling with Polar Coordinates

Multidimensional scaling suffers from several problems. Besides the quadratic need of memory, MDS, as described above is solved by an iterative method, expensive with respect to computation time. Furthermore, a completely new solution must be recalculated, if a new object is added to the dataset.

For a p-dimensional dataset X MDS$_{polar}$ determines a 2-dimensional representation in polar coordinates $Y = \{(l_1, \varphi_1), \ldots, (l_n, \varphi_n)\}$, where the length l_k of the original vector x_k is preserved and only the angle φ_k has to be optimised. This solution is defined to be optimal, if all angles between pairs of data objects in the projected dataset Y coincide as good as possible with the angles in the original feature space X.

The use of polar coordinates in the space Y preserving the length of each feature vector has various advantages. On the one hand, the number of parameters to be optimised is reduced, while the length preseservation already guarantees a roughly correct placement of the feature vectors in the lower dimensional space. On the other hand, this property will enable us to reduce the complexity of the algorithm based on the following idea. It is important to map feature vectors that are close to each other in the original space X to close vectors Y, whereas feature vectors with a large distance between them should also be mapped far away from each other. However, for vectors with a large distance, it is not important that we match the original distance exactly, but it is sufficient to make sure that they will not be mapped close to each other. When the length of two vectors differs significantly, we already guarantee a certain distance between the projected vectors, even if the angle between them is not matched at all. Using this property it will not be necessary to consider the angles between all vectors. For those vectors with a significant difference in length, we can neglect the angle.

A straight forward definition of an objective function to be minimised for this problem, taking all angles into account for the moment, would be

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (|\varphi_i - \varphi_k| - \psi_{ik})^2 \tag{1}$$

where φ_k is the angle of y_k, ψ_{ik} is the positive angle between x_i and x_k, $0 \leq \psi_{ik} \leq \pi$. E is minimal, if the difference of the angle of each pair of vectors of dataset X and the corresponding two vectors in dataset Y is zero. The absolute value is chosen in equation (1) because the order of the minuends can have an influence on the sign of the resulting angle. As discussed in [8] equation (1) is not suitable for finding an analytical solution or a gradient descent technique, since functional E is not differentiable.

In the mentioned paper it is proposed to minimise the following functional instead

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - \psi_{ik})^2. \tag{2}$$

The problem, that arises with equation (2) is, that a simple minimization would not lead to acceptable results. This is the case because an angle between y_i and y_k, that might perfectly match the angle ψ_{ik}, $\varphi_i - \varphi_k$ can either be ψ_{ik} or $-\psi_{ik}$. Therefore, when minimising functional (2) in order to actually minimise functional (1), one can take the freedom to choose whether we want the term $\varphi_i - \varphi_k$ or the term $\varphi_k - \varphi_i$ to appear in (2).

When we are free to choose between $\varphi_i - \varphi_k$ and $\varphi_k - \varphi_i$ in equation (2), we take the following into account

$$(\varphi_k - \varphi_i - \psi_{ik})^2 = (-(\varphi_k - \varphi_i - \psi_{ik}))^2 = (\varphi_i - \varphi_k + \psi_{ik})^2.$$

Therefore, instead of exchanging the order of φ_i and φ_k, one can choose the sign of ψ_{ik}, leading to

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - a_{ik}\psi_{ik})^2 \tag{3}$$

with $a_{ik} = \{-1, 1\}$. In order to solve this modified optimization problem of equation (3) we take the partial derivatives of E, yielding

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - a_{ik}\psi_{ik}). \tag{4}$$

To fulfil the necessary condition for a minimum one sets equation (4) equal to zero and solves it for the φ_k-values, which leads to

$$\varphi_k = \frac{\sum_{i=1}^{k-1} \varphi_i - \sum_{i=1}^{k-1} a_{ik}\psi_{ik}}{k-1}. \tag{5}$$

Thus, on the one hand, neglecting that we still have to choose a_{ik}, our solution is described by a system of linear equations which means its solution can be calculated directly without the need of any iteration procedure. On the other hand, as described above, one has to handle the problem of determining the sign of the ψ_{ik} in the form of the a_{ik}-values. A greedy strategy with $O(n \cdot \log n)$ for this is also proposed in [8].

IV. POLARMAP

As an extension of MDS$_{polar}$ we propose in this work a method that learns a function f that provides for any p-dimensional feature vector x_k the corresponding angle φ_k that is needed to map the feature vector to a 2-dimensional feature space. As for MDS$_{polar}$ the length of vector x_k is preserved. With the obtained function also angles for new feature vectors can be computed. A 2-dimensional scatter plot might be not suitable, when visualising mappings for large datasets. With

the computed function it is simple to produce information murals, which allow more comprehensive visualizations [4].

Analogous to functional (1) we define our objective function E as follows:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(|f(x_i) - f(x_j)| - \psi_{ij}\right)^2. \tag{6}$$

E is minimal, if, for each pair of feature vectors, the difference of the two angles, which are computed by the respective function f is equal to the measured angle of the two vectors in the original space. Since functional (6) is not differentiable, we propose analogous to the procedure for MDS_{polar} to minimise the following differentiable objective function

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(f(x_i) - f(x_j) - \psi_{ij}\right)^2. \tag{7}$$

Since according to our definition $\psi_{ij} \geq 0$, it is obvious that $E_{min} \leq \tilde{E}_{min}$. Thus, a minimum of $\tilde{E}$ might not be the minimum of E, but it can be used as a conservative estimation. Albeit, f might be any function, we discuss in this work the following function style

$$f(x) = a^T \cdot \tilde{x}, \tag{8}$$

where a is vector whose components are the parameters to be optimised and $\tilde{x}$ is feature vector x itself or a modification of x. In the simplest case we use

$$\begin{aligned} \tilde{x} &= x \\ a &= (a_1, a_2, \ldots, a_p)^T \end{aligned} \tag{9}$$

where f describes in fact the linear combination of x.

Assuming that a certain component of x affects the transformation not linearly but quadratically or exponentially, it may be useful to compute some additional components from x with the objective to gain more coefficients, which could improve the transformation.

An example for quadratic components derived from x is described by the following choice:

$$\begin{aligned} \tilde{x} &= (x_1, \ldots, x_p, x_1x_1, \ldots, x_1x_p, \\ &\quad x_2x_2, \ldots, x_2x_p, \ldots, x_px_p)^T \qquad (10) \\ a &= (a_1, \ldots, a_p, a_{11}, \ldots, a_{1p}, \\ &\quad a_{22}, \ldots, a_{2p}, \ldots, a_{pp})^T. \qquad (11) \end{aligned}$$

Replacing term f by the respective function we obtain

$$\begin{aligned} \tilde{E} &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T\tilde{x}_i - a^T\tilde{x}_j - \psi_{ij}\right)^2 \qquad (12) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T(\tilde{x}_i - \tilde{x}_j) - \psi_{ij}\right)^2. \qquad (13) \end{aligned}$$

For a better readability we replace $\tilde{x}_i - \tilde{x}_j$ by $\tilde{x}_{ij}$ and obtain

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T\tilde{x}_{ij} - \psi_{ij}\right)^2. \tag{14}$$

Algorithm 1 **Greedy POLARMAP**

$\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \quad \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      if $\theta_{ij}a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
      end if
    end for
  end for
  $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
  compute $\tilde{E}$
until $E' \leq \tilde{E}$

The derivative of $\tilde{E}$ w.r.t. a can by easily obtained

$$\frac{\partial\tilde{E}}{\partial a} = 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij}. \tag{15}$$

Setting equation (15) equal to zero to fulfil the necessary condition for a minimum we end up with

$$0 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} \tag{16}$$

which results in a system of linear equations in $a = (a_1, a_2, \ldots, a_p)^T$.

As mentioned already, angles computed by $f(x_i) - f(x_j)$, might be positive or negative, while ψ_{ij} is always positive by definition. Thus, in the case where $a^T\tilde{x}_{ij} < 0$ holds, $\tilde{E}$ might be minimal, but our original objective function E might not be minimal. Hence, replacing $\tilde{x}_{ij}$ by $-\tilde{x}_{ij}$ in this case might lower the error. Consequently, finding the appropriate sign for $\tilde{x}_{ij}$ is a crucial step when minimising $\tilde{E}$. For common datasets determining the exact solution for this problem is too expensive regarding computation time. In the following section we describe a greedy strategy that approximates a relaxation of this problem.

A. A Greedy Algorithm for the Approximation of POLARMAP

Determining the sign for each $\tilde{x}_{ij}$ requires exponential need of computation time in the number of feature vectors. For real-world datasets this is unacceptable. When relaxing the problem in favour to an approximation of the exact solution one can reduce the time complexity down to $O(n\cdot\log n)$. In this section we begin with a very fast greedy algorithm that finds rather poor approximations of the exact solution, which are suitable for initialization purposes for more complex approximation schemes.

In the following, we use a $n \times n$-matrix Θ with $\theta_{ij} = 0$ when the sign for $\tilde{x}_{ij}$ is positive and $\theta_{ij} = 1$ when the sign for $\tilde{x}_{ij}$ is negative. As algorithm (1) shows, this greedy algorithm changes first these signs for the respective $\tilde{x}_{ij}$ when $\theta_{ij}a^T\tilde{x}_{ij} < 0$ is satisfied and computes afterwards the updated components of a by solving the revised system of

Algorithm 2 **Greedy POLARMAP**

$\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \quad \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      if $\theta_{ij}a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
        $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
        compute $\tilde{E}$
        GOTO: check
      end if
    end for
  end for
  LABEL: check
until $\tilde{E}' \leq \tilde{E}$

linear equations. Usually, this algorithm converges after a few iterations. This approach is very efficient and simple at the same time.

Algorithm (2) shows another greedy algorithm. Always, when θ_{ij} changes, a will be recomputed immediately and the next iteration starts. Θ changes during one iteration at most in one point – namely θ_{ij}, otherwise the algorithm ends without changing any θ. Thus, the algorithm greedily changes the first θ_{ij}, when condition $\theta_{ij}a^T\tilde{x}_{ij} < 0$ is satisfied. From this it follows that the algorithm only finds a local minimum of $\tilde{E}$, which is the reason why we speak about a relaxation of the problem.

It is advisable to initialise Θ with algorithm (1). Otherwise, lots of iterations will be needed until convergence. With algorithm (2) very accurate transformations will be found – indeed computational costs are fairly high. In the following subsection, we describe a technique that reduces the computation cost drastically.

B. Generalization of POLARMAP

Although the above greedy algorithm is efficient, for large datasets too many iterations will be needed until convergence. Its time and space complexity are also quadratic in the number of data, so that it is not applicable to larger datasets. In order to evaluate the objective function, all $\tilde{x}_{ij}$- and all ψ_{ij}-values must be computed in advance, causing already the quadratic complexity. The greedy algorithm must also compute many (again quadratic in the number of data) scalar products $a^T\tilde{x}_{ij}$ that contribute not or only little to the quality of the transformation. Thus, if we had a measure to decide whether an adaptation of θ_{ij} might be target-oriented or not, we could save computation time by skipping the computation of nonessential scalar products.

As a matter of fact, the computation of the error $\tilde{E}$ accounts for the greatest part of the computation resources. In the following we will discuss the problem how to reduce computation time due to dropping hopefully dispensable terms.

As for MDS$_{polar}$ we provide for POLARMAP a generalization by introducing weights w_{ij} for our objective function $\tilde{E}$, that results in

$$\tilde{E} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)^2. \tag{17}$$

Again, we obtain the following system of linear equations after taking partial derivatives of $\tilde{E}$

$$\frac{\partial\tilde{E}}{\partial a} = 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} \tag{18}$$

and setting equation (18) to zero to fulfil the necessary condition for a minimum which leads to

$$0 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij}. \tag{19}$$

The introduction of weights opens new ways to define and handle the objective function. We cannot only assign a weight to individual errors, controlling in this way how much influence single errors have on the final result. We can also consider relative instead of absolute errors. For example, choosing $w_{ij} = 1/\psi_{ij}^2$ corresponds to relative MDS$_{polar}$. The difference between the angle in the original space and the corresponding angle in the target space, will not account directly to the computation of a but weighted with ψ_{ij}.

Weights can be chosen in such a way, that only feature vectors, which are similar to a certain degree will be taken into account, when computing θ_{ij}.

Since our transformation preserves the length of each data vector, it is guaranteed that vectors with a large difference in length will not be mapped to close points in the plane, even though their angle might not be matched at all. Therefore, we propose to use a small or even zero-weight for pairs of data vectors that differ significantly in their length. The weight could be defined as a function of the difference between the length values l_i and l_j of two data vectors:

$$w_{ij} = w(l_i, l_j) = w(z). \tag{20}$$

We can use the absolute difference for z, i.e.

$$z = z_a = |l_i - l_j|.$$

This might be useful if certain information about the structure of the data is known in advance. The argument z_r for relative weighting functions

$$z = z_r = \min\left\{\frac{l_i}{l_j}, \frac{l_j}{l_i}\right\}$$

might be useful if a certain threshold can be provided. To decrease the computational complexity, weights should be chosen in such a way, that for feature vectors with a certain (large) distance the respective weights become zero. The following function describes a simple weighting function that behaves as just mentioned:

$$w(z_r) \quad = \quad \begin{cases} \sqrt{\left(\frac{z_r - \vartheta}{1-\vartheta}\right)} & \text{, if} \quad z_r \geq \vartheta \\ 0 & \text{, otherwise} \end{cases} \tag{21}$$
$$\text{where} \quad \vartheta \in [0, 1].$$

Algorithm 3 **Greedy POLARMAP**

$\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
sort X
set $maxbinsize$
initialise n_i
let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \quad \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n_i} w_{ij}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n_i **do**
      $w_{ij} \leftarrow w(l_i, l_j)$
      if $w_{ij} > c$ **then**
        $n_i \leftarrow j - 1$
        GOTO: check1
      end if
      if $\theta_{ij} a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
        $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n_i} w_{ij}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
        compute $\tilde{E}$
        GOTO: check
      end if
    end for
    LABEL: check1
  end for
  LABEL: check
until $\tilde{E}' \leq \tilde{E}$

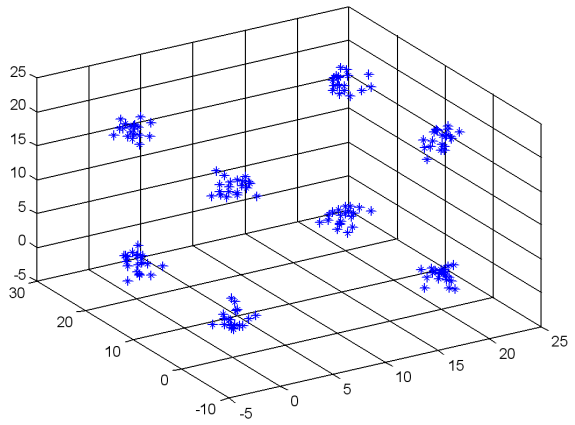
With the threshold ϑ one can control indirectly the fraction of the data, that will be used to affect a. Thus, small values for ϑ lead to many non-zero weights, which comes along with high computational complexity. Values near 1 for ϑ lead to a quickly decreasing weighting function and to low computational complexity, respectively. Any other function can be used as weighting function.

For reasons of an easy implementation and low computational complexity a decreasing function which leads to a more or less large fraction of zero weights should be used. For an efficient implementation it is useful to sort the feature vectors by means of their length. Note that this can be achieved with $O(n \cdot \log n)$ time complexity.

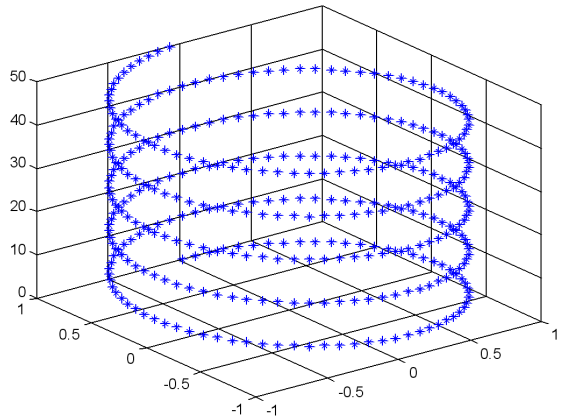
It is obvious, when the feature vectors are sorted in that way, weights for a given i are decreasing while incrementing j because the length of $\tilde{x}_{ij}$ is increasing. Therefore, the inner for-loop in algorithm (2) can be interrupted if $w(z)$ becomes zero for the first time. Since the weighting function is decreasing, a further iteration would lead to zero, too. In cases where clusters with a large amount of data are expected in a dataset, it might be rather useful to limit the maximum number of pairs to consider for the calculation of a and θ, than setting a larger threshold. It might also be useful to reduce ϑ locally, when only few vectors satisfy the condition in equation (21).

With a limitation of the number of weights $w > 0$ and a moderate ϑ at the same time, it can be achieved that the number of weights considered for the calculation of φ_k does not differ too much for different φ_k and limited computation time can be guaranteed.

Algorithm (3) is a modification of the previous algorithm regarding the mentioned aspects. Beneath sorting X, the inner for-loop now contains the condition to break the loop if the weighting function indicates, that for the given i no further $\tilde{x}_{ij}$ has to be considered.



(a) Cube Dataset



(b) Coil Dataset

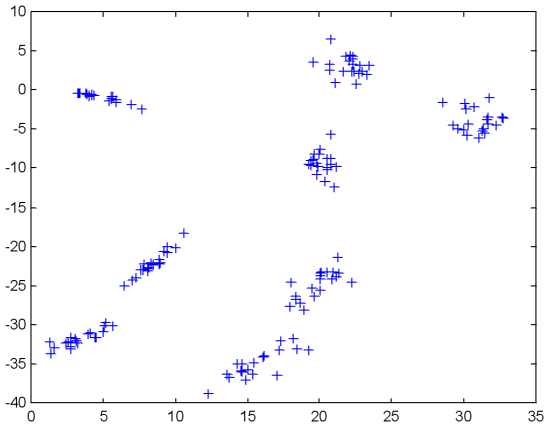
Fig. 1. Synthetic Datasets

Note, that we reduce computation time drastically, if we choose an appropriate weighting function. Instead of $O(c \cdot n^2)$ with c as number of iterations, we obtain $O(c \cdot n \cdot m)$ where m is the maximum bin size. The bin size for a feature vector x_i refers to the number of non-zero weights w_{ij}. With this bin-strategy we do not only reduce the number of pairs $\tilde{x}_{ij}$ to be considered, much more important is the effect on the computation of a and $\tilde{E}$. With algorithm (3) we introduced the array $n_i,\ i = 1 \ldots n$, that is initialised with $n_i = \min(i + maxbinsize, n),\ \forall i$. When computing a and $\tilde{E}$, it is no longer necessary to sum up the difference between the target angle and ψ_{ij} for vectors which are out of the bin, since it will be weigthed with w_{ij} which is zero.

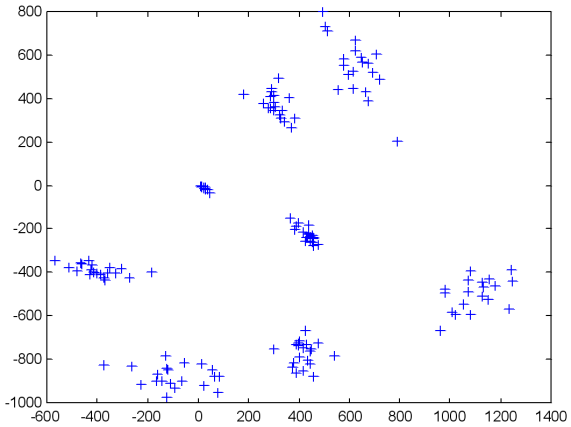
Of course, if the bin size is only limited by a weighting function that leads only to few weights $w_{ij} = 0$, the gain of computation time tends to zero.

V. Results

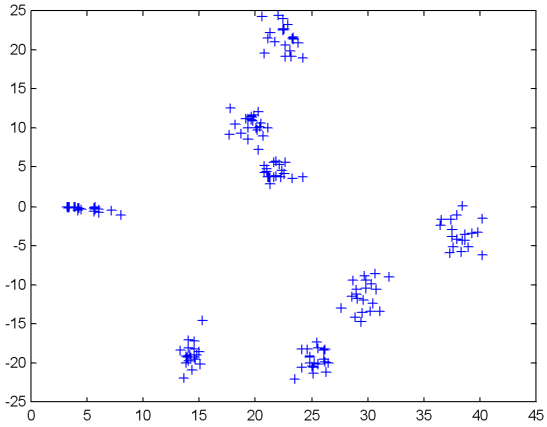
In this section, we discuss the results of POLARMAP on two synthetic 3-dimensional datasets. Figure 1 shows these



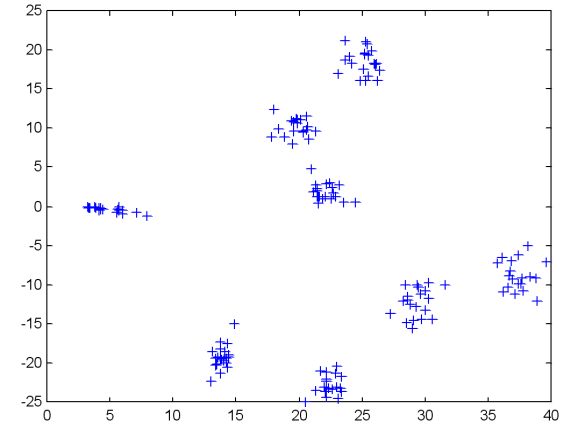
(a) Algorithm 1 ($\tilde{x} = x$)



(b) Algorithm 1 ($\tilde{x}$ according to eq.(10))



(c) Algorithm 2 ($\tilde{x} = x$)



(d) Algorithm 3 ($\tilde{x} = x$)

Fig. 2. Results on Cube Dataset

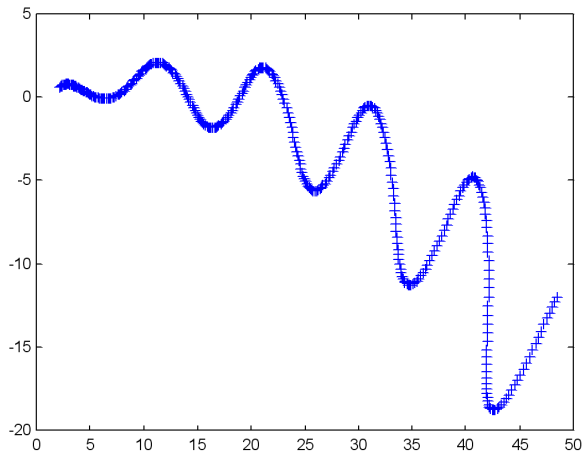
datasets. The cube dataset (a) is about a dataset, where data points scatter around the corners of an imaginary 3-dimensional cube. Thus, the cube dataset contains eight well separated clusters. The coil dataset (b) contains a 1-dimensional manifold comparable to a serpentine. Furthermore, we apply POLARMAP on the well known iris dataset. A Sammon mapping of the 4-dimensional iris dataset is shown in figure 4 (a). We split this dataset into a training dataset and a test dataset to demonstrate the capability POLARMAP to generalise. The wine dataset results from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. A Sammon mapping of the 13-dimensional wine dataset is shown in figure 4 (c).

Our tests have shown, that algorithm (1) is a good initialization for algorithm (2) and (3). Since algorithm (2) and (3) compute the actual coefficients immediately after changing one sign, without initialization, many iterations would be needed for large datasets until convergence. For that reason it is advisable to initialise algorithm (2) and (3) with algorithm (1). Note that algorithm (1) does not have quadratic complexity, when we introduce corresponding weights leading to moderate bin sizes. The following transformations result from this procedure.

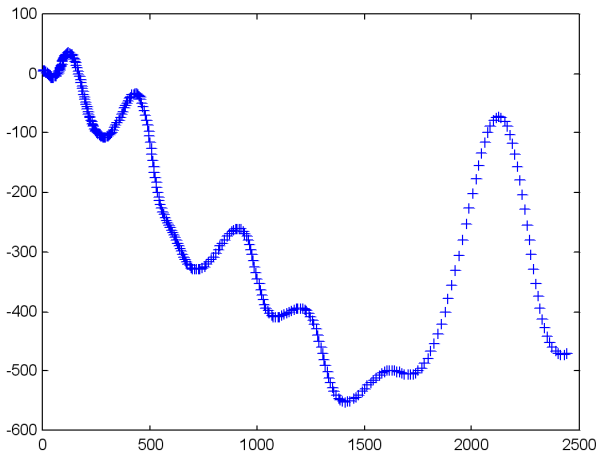
Figure 2 shows some results on the cube dataset. The greedy algorithm (1) converges already after three iterations when using $\tilde{x}$ according to equation (9) (subfigure (a)). Similarly, greedy algorithm (1) converges after five iterations, when using $\tilde{x}$ according to equation (10) (subfigure (b)). For the relatively simple cube dataset it is not of much importance, to generate additional components for $\tilde{x}$ and additional components a respectively. Subfigure (c) and (d) result from applying algorithm (2) and algorithm (3) respectively. These transformations are based on the $\tilde{x} = x$ cube dataset. Obviously, a linear function with three coefficients a is sufficient to map the cube dataset to a 2-dimensional feature space. The eight clusters are clearly separated in the target space, too. Using weights according to algorithm (3) and the following weighting function

$$w(z_a) = \begin{cases} 1 & \text{if } z_a < c \\ 0 & \text{otherwise} \end{cases}$$

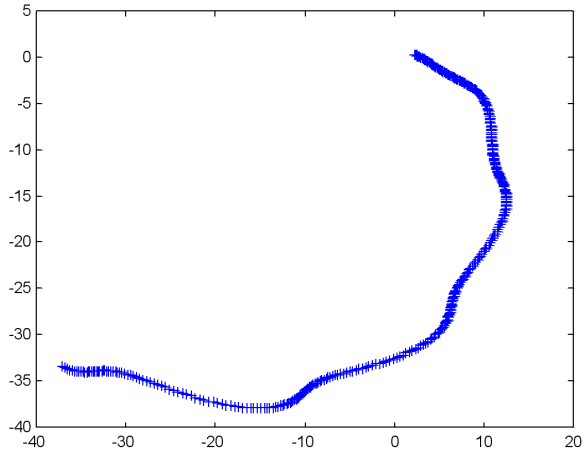
with $c = 35$ leads to the transformation shown in subfigure (d). The results with algorithm (2) and algorithm (3) are quite similar for the cube dataset, even though algorithm (3) needs less computation time.



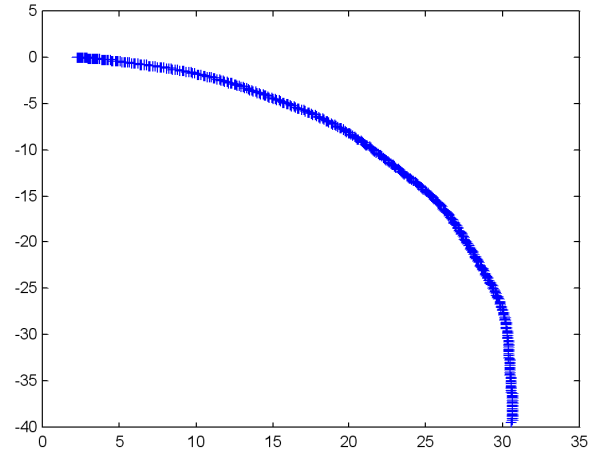
(a) Coil Dataset, Greedy 1



(b) Coil Dataset, Greedy 1, eq. (10)



(c) Coil Dataset, Bins, absolute



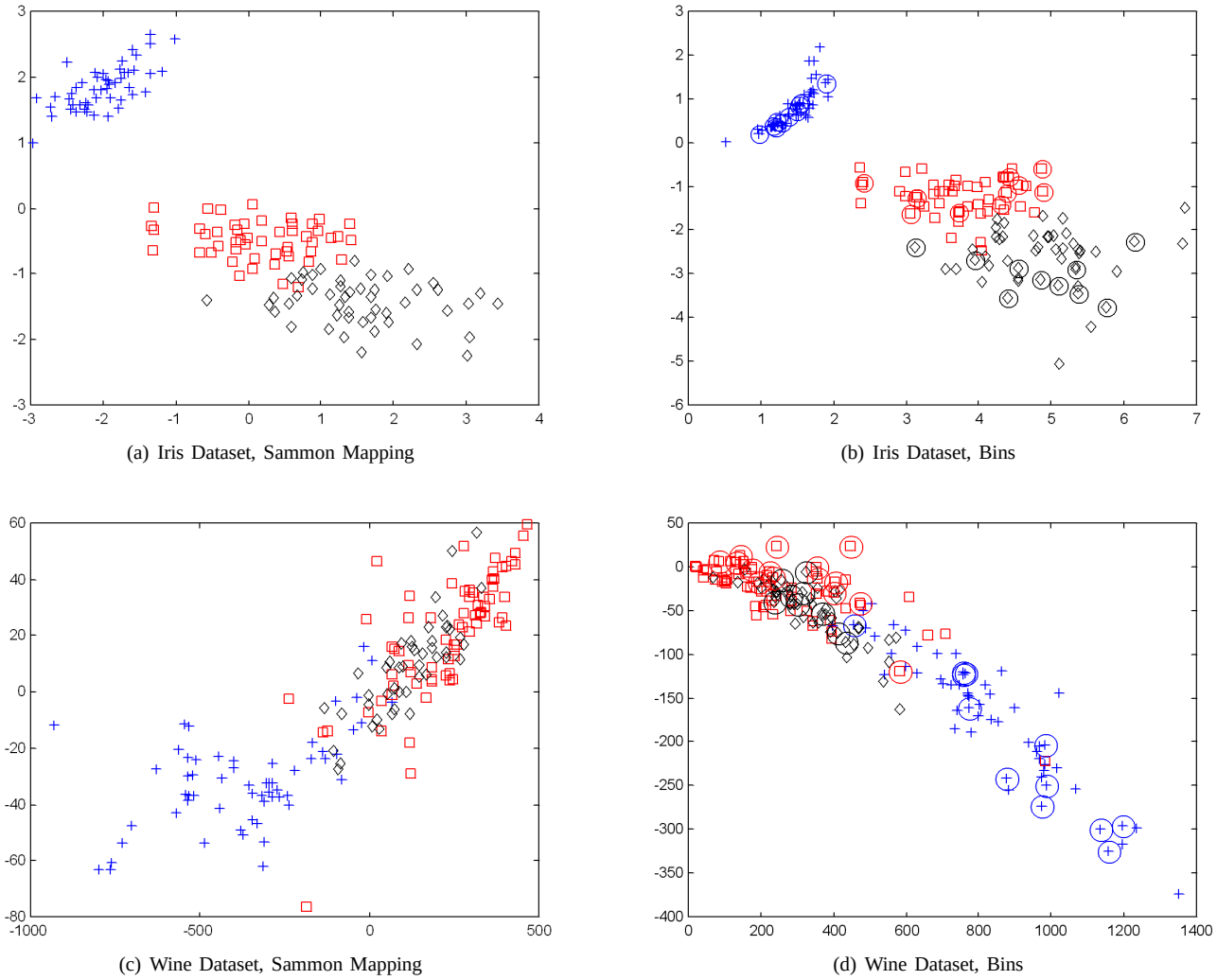
(d) Coil Dataset, Bins, relative

Fig. 3. Results on Coil Dataset

The results for the coil dataset are shown in figure 3. Again, subfigure (a) results from algorithm (1) with $\tilde{x} = x$ and subfigure (b) results from $\tilde{x}$ according to equation (10). The results for both representations of the dataset are similar regarding a majority of the characteristics. Algorithm (1) converges after few iterations for both datasets. Subfigure (c) and (d) show the results of algorithm (3) on the dataset ($\tilde{x} = x$) without initialising Θ other than zero. Subfigure (c) results from using a bin size of 10. Subfigure (d) results from using $w_{ij} = 1/\psi_{ij}^2$ for all w_{ij} inside the bin of size 10. Chosing maximum bin size 10 leads to the fact, that no sign will be changed and thus the algorithm stops after one iteration. Designing the weighting function such that a larger bin size has to be taken into account, one can observe that already after only three signs have changed, the transformation gets the major characteristics as the one in subfigure (a). Even if this transformation is very simple, some requirements, i.e. preserving distances between feature vectors, are fulfilled.

Since a function is learned by POLARMAP it becomes possible to map new vectors in the target space. To demonstrate the power of POLARMAP, we applied it on the well known iris dataset. Figure 4 (a) shows the Sammon mapping of the iris dataset. The different classes are represented by different symbols. Subfigure (b) shows the transformation with POLARMAP. For this example, the iris dataset is split into a training dataset and a test dataset. The training dataset consists of 80% of each class. This part of the data is used to learn the desired coefficients. The test dataset, that contains the remaining 20% of the data, is mapped to the target space by means of the learned function. The mapping of the training dataset is plotted with the different symbols again, each for the corresponding class. The mapped feature vectors of the test dataset are additionally marked with a circle around the corresponding symbol. As the figure shows, the learned function maps the feature vectors in the right way.

Figure 4 (c) shows the Sammon mapping of the wine dataset. The three classes are marked with different symbols again (class 1: $+$, class 2: $\square$, class 3: $\diamond$!). Based on the Sammon mapping, the three classes cannot be separated linearly. Notably class 2 and class 3 cannot be distinguished. The transformation of the wine dataset with POLARMAP is shown in figure 4 (d). Both transformations are similar



(a) Iris Dataset, Sammon Mapping

(b) Iris Dataset, Bins

(c) Wine Dataset, Sammon Mapping

(d) Wine Dataset, Bins

Fig. 4. Results on Iris Dataset and Wine Dataset

regarding the scattering of the different classes. The mapping of the new feature vectors (marked with a circle around the respective symbol) meets the expectations from the mapping of the training dataset.

Figure 5 shows the effect of the bin size on the transformation accuracy according to the POLARMAP criterion (solid line) and the Sammon criterion (dashed line) on the wine dataset. Both measures indicate a better mapping with increasing bin size at the beginning. This is what we expect indeed. For larger bin sizes the error is increasing slightly again. The probability to get stuck in a local minimum seems to increase with larger bin sizes. As the figure reveals as well, the error is not decreasing linearly. Thus, using the bin technique, the user has to make the compromise between transformation quality and computation/space complexity. In many cases it may be sufficient to use a small bin size to get an overall view of the data. As for the wine dataset, the error can be reduced drastically, investing resources in the consideration of a higher bin size.

VI. Conclusion

In this paper we have presented a powerful data visualization method. Under the constraint to preserve the length of feature vectors, it was our aim to find a mapping that projects feature vectors from a high-dimensional to the plane in such a way that we minimise the errors in the angles between the mapped feature vectors. The solution is described by a system of linear equations. To overcome the problem in high-dimensional feature spaces, that no differentiation between positive and negative angles can be made as for a 2-dimensional feature space, three algorithms are provided to obtain the desired signs for the angles. With the bin-algorithm, we presented an algorithm, that lowers the computation complexity down to $O(n \cdot \log n)$.

With the function, learned by means of POLARMAP it is possible to map even new feature vectors to the target space without any extra costs. Experimental results demonstrate the power of the approach. For very large datasets, it is also possible to construct a mapping from a sample of a reasonable size with acceptable computation time and then project the whole datasets using the constructed mapping. For

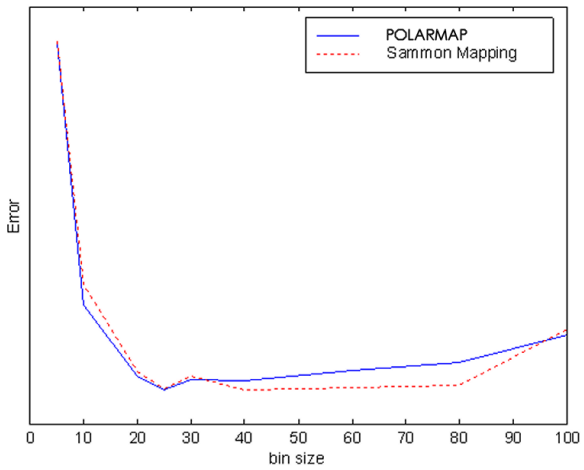


Fig. 5. The Effect of the Bin Size

large datasets it is also recommended, not to display them as a scatter plot with a dot or symbol for each feature vectors, but to use density-based scatter plots [4] that our method also supports.

REFERENCES

[1] Borg, I., Groenen, P.: Modern Multidimensional Scaling : Theory and Applications. Springer, Berlin (1997).
[2] Chalmers, M.: A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data. Proceedings of IEEE Visualization 1996, San Francisco, CA (1996), 127–132.
[3] Faloutsos, C., Lin, K.: Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. Proceedings of ACM SIGMOD International Conference on Management of Data, San Jose, CA (1995), 163–174.
[4] Jerding, D.F., Stasko, J.T.: The information mural: a technique for displaying and navigating large information spaces. Proceedings of the 1995 IEEE Symposium on Information Visualization, (1995), 43–50.
[5] Kruskal, J.B., Wish, M.: Multidimensional Scaling. SAGE Publications, Beverly Hills (1978).
[6] Lowe, D., Tipping, M.E.: Feed-Forward Neural Networks Topographic Mapping for Exploratory Data Analysis. Neural Computing and Applications, 4, (1996), 83–95.
[7] Morrison, A., Ross, G., Chalmers, M.: Fast Multidimensional Scaling through Sampling, Springs and Interpolation. Information Visualization (2003) 2, 68–77.
[8] Rehm, F., Klawonn, F., Kruse, R., MDS_{polar} - A New Approach for Dimension Reduction to Visualize High Dimensional Data. In: Advances in Intelligent Data Analysis VI: 6th International Symposium on Intelligent Data Analysis, IDA 2005, Springer, (2005), 316–327.
[9] Tenenbaum, J.B., de Silva, V., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science, 290, (2000), 2319–2323.
[10] Williams, M., Munzner, T.: Steerable, Progressive Multidimensional Scaling. 10th IEEE Symposium on Information Visualization, Austin, TX (2004), 57–64.