MDS$_{polar}$: A New Approach for Dimension Reduction to Visualize High Dimensional Data

Frank Rehm[1], Frank Klawonn[2], and Rudolf Kruse[3]

[1] German Aerospace Center, Braunschweig, Germany
[2] University of Applied Sciences, Braunschweig/Wolfenbuettel, Germany
[3] Otto-von-Guericke-University of Magdeburg, Germany

Abstract. Many applications in science and business such as signal analysis or costumer segmentation deal with large amounts of data which are usually high dimensional in the feature space. As a part of preprocessing and exploratory data analysis, visualization of the data helps to decide which kind of method probably leads to good results. Since the visual assessment of a feature space that has more than three dimensions is not possible, it becomes necessary to find an appropriate visualization scheme for such datasets. In this paper we present a new approach for dimension reduction to visualize high dimensional data. Our algorithm transforms high dimensional feature vectors into two-dimensional feature vectors under the constraints that the length of each vector is preserved and that the angles between vectors approximate the corresponding angles in the high dimensional space as good as possible, enabling us to come up with an efficient computing scheme.

1 Introduction

Many applications in science and business such as signal analysis or costumer segmentation deal with large amounts of data which are usually high dimensional in the feature space.

Before further analysis or processing of data is carried out with more sophisticated data mining techniques, data preprocessing and exploratory data analysis is an important step. As a part of this process, visualization of the data helps to decide which kind of method probably leads to good results. Since the visual assessment of a feature space that has more than three dimensions is not possible, it becomes necessary to find an appropriate visualization scheme for such datasets.

The general data visualization problem we consider here is to map high dimensional data to a two-dimensional plane – usually a computer screen – trying to preserve as many properties or as much information of the high dimensional data as possible. In other words, we have to face a dimension reduction problem. A very simple approach is to look at scatter plots obtained from projections to two selected features. However, each scatter plot will only contain the information of the two chosen features and with a high number of features it is infeasible to inspect the scatter plots resulting from all possible combinations of two features. Before finding a mapping of the high dimensional data to the two-dimensional plane, an error or quality measure must be defined in order to evaluate the suitability of such a mapping. Principal component analysis is one possible choice, producing an affine transform that preserves as much of the variance in the data as possible. However, instead of the variance, other criteria like the distance between the single data vectors might be of higher interest. Multidimensional scaling (see for instance [1, 4]) is a technique that aims at preserving the distances between the data, when mapping them to lower dimensions. Although multidimensional scaling and related approaches yield promising and interesting results, they suffer from high computational needs concerning memory as well as computation time. In recent years some research has been done in this regard [2, 6, 7].

In this paper we present a new approach for dimension reduction to visualize high dimensional data. Instead of trying to preserve the distances between feature vectors directly, our algorithm transforms high dimensional feature vectors into two-dimensional feature vectors under the constraints that the length of each vector is preserved and that the angles between vectors approximate the corresponding angles in the high dimensional space as good as possible, enabling us to come up with an efficient computing scheme. After a brief review of the concept of multidimensional scaling and related approaches, we explain the theoretical background of our approach and discuss some illustrative examples in the end of the paper.

2 Multidimensional Scaling

Multidimensional scaling (MDS) is a method that estimates the coordinates of a set of objects $Y = \{y_1, \dots, y_n\}$ in a feature space of specified (low) dimensionality that come from data $X = \{x_1, \dots, x_n\} \subset \Re^p$ 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

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

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 $D^y = \left(d^y_{ij}\right)$, $d^y_{ij} = \|y_i - y_j\|$, $i, j = 1, \dots, n$ of the corresponding transformed dataset will be minimized.

Thus, different error measures to be minimized 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*

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

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

$$\frac{\partial E}{\partial y_k} = \frac{2}{\sum\limits_{i=1}^{n} \sum\limits_{j=i+1}^{n} d_{ij}^x} \sum_{j \neq k} \frac{d_{kj}^y - d_{kj}^x}{d_{kj}^x} \frac{y_k - y_j}{d_{kj}^y}.$$

After random initialization, for each projected feature vector y_k a gradient descent is carried out and the distances d_{ij}^y as well as the gradients $\frac{\partial d_{ij}^y}{\partial y_k}$ 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 into the lower dimensional space, but the whole MDS procedure has to be repeated. NeuroScale [5] 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 this paper, we propose a different algorithm, not needing any iterative scheme and whose complexity can be reduced to $O(n \cdot \log n)$.

3 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 calculated, if a new object is added to the dataset.

With MDS$_{polar}$ we present a new approach to find a two-dimensional projection of a p-dimensional dataset X. MDS$_{polar}$ tries to find a 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 optimized. Thus, our 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.

A straight forward definition of an objective function to be minimized for this problem 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 all pairs of vectors of dataset X and the corresponding two vectors in dataset Y are 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. The problem with this notation is that the functional E is not differentiable, exactly in those points we are interested in, namely, where the difference between angles φ_i and φ_k becomes zero. Another intuitive approach would be

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

In this case the derivative can be determined easily, however, resulting in a system of nonlinear equations for which no analytical solution can be provided.

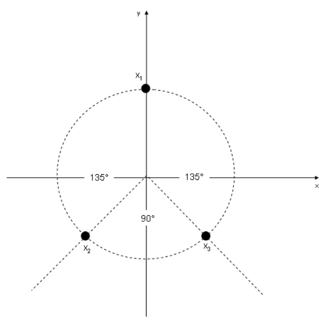
In order to overcome these difficulties, we propose an efficient method that enables us to compute an approximate solution for a minimum of the objective function (1) and related ones. In a first step we ignore the absolute value in (1) and consider

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

instead. When we simply minimize (3), the results will not be acceptable. Although the angle between y_i and y_k might perfectly match the angle ψ_{ik}, $\varphi_i - \varphi_k$ can either be ψ_{ik} or $-\psi_{ik}$. Since we assume that $0 \leq \psi_{ik}$ holds, we always have $(|\varphi_i - \varphi_k| - \psi_{ik})^2 \leq (\varphi_i - \varphi_k - \psi_{ik})^2$. Therefore, finding a minimum of (3) means that this is an upper bound for the minimum of (1). Therefore, when we minimize (3) in order to actually minimize (1), we 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 (3). Before we discuss techniques to minimize (3) with the freedom of reordering, we have to preprocess the data in order to fit them best to our approach.

3.1 Data Preprocessing

The following figure illustrates an important problem by means of a simple dataset. The table next to the graphics contains the values of the angles between all three feature vectors.



ψ	x_1	x_2	x_3
x_1	0	135	135
x_2	135	0	90
x_3	135	90	0

Even though, this feature space has only two dimensions and therefore an exact reproduction of the dataset should be possible, this cannot be achieved without additional preprocessing. Since we only want to preserve the angles between data vectors, it is obvious that any solution will be invariant with respect to rotation of the dataset. Thus, assuming without loss of generality $\varphi_1 = 0$ enforcing $\varphi_2 = 135$, then according to our objective function (1) $\varphi_3 = 180$ leads to the optimal solution, which is obviously not what we are looking for. This problem is caused by the fact that ψ_{ik} is defined as a positive angle which satisfies $\psi_{ik} \leq 180°$. This problem can be solved easily by translating all feature vectors into the first quadrant. More generally, for a higher dimensional dataset we apply a translation that makes all components of data vectors non-negative. For this we only have to determine for each component the largest negative value occurring in the dataset and using this as a positive value of the corresponding component of the translation vector. Note that, when the dataset is normalized, i.e. all components are between 0 and 1, no further preprocessing is required.

Thus, doing this kind of preprocessing, we actually do not preserve the original data properties but those after the transformation. Of course, rotation and translation is not changing any inter-data properties. The translation vector has to be stored so that for incremental adding of new objects the transformation can be performed accordingly. For most of the new objects the transformation will be as requested. It may occur that for new objects which have one or more extreme components the translation will not be sufficient to eliminate the negative components. In such a case, which is rather rare if the previous data is representative, the mapping of the respective object is still working, but not that exact sometimes.

3.2 Approximation of MDS$_{polar}$

When we are free to choose between $\varphi_i - \varphi_k$ and $\varphi_k - \varphi_i$ in (3), 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, we can choose the sign of ψ_{ik}, leading to

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

with $a_{ik} = \{-1, 1\}$. In order to solve this modified optimization problem of equation (4) we take the partial derivatives of E, yielding

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - a_{ik}\psi_{ik}). \tag{5}$$

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, we have to handle the problem of determining the sign of the ψ_{ik} in the form of the a_{ik}-values.

To fulfil the necessary condition for a minimum we set equation (5) equal to zero and solve for the φ_k-values, which leads to

$$\varphi_k = \frac{\sum_{i=1}^{k-1} \varphi_i - \sum_{i=1}^{k-1} a_{ik}\psi_{ik}}{k-1}. \tag{6}$$

Different optimization strategies are conceivable. Of course, an important condition is the computational complexity of the respective approximation algorithm. In this paper we present a number of different strategies, starting with a greedy algorithm which is quadratic with the number of data objects in time, but is linear in space. Later on, we propose an algorithm that can even reduce the complexity to $O(n \cdot \log n)$.

3.3 A Greedy Algorithm for the Approximation of MDS$_{polar}$

As mentioned above, this solution describes a system of linear equations. Since the desired transformation is rotation invariant φ_1 can be set to any value, i.e. $\varphi_1 = 0$. By means of a greedy algorithm we choose $a_{ik} \in \{-1, 1\}$ such that for the resulting φ_k the error E of the objective function (4) is minimal. For φ_2 the exact solution can always be found, since a_{12} is the only parameter to optimize. For the remaining φ_k the greedy algorithm sets a_{ik} in turn either -1 or 1, verifying the validity of the result, setting a_{ik} the better value immediately and continuing with the next a_{ik} until all $k-1$ values for a_{ik} are set.

Algorithm 1 describes in a simplified way the greedy method. When implementing the method, it can be optimized in that way, that the first φ_k in the *for*-loop has not always to be recalculated if in step $i-1$ the parameter a_{ik} has not been changed to -1. In such cases φ_k holds the value from the previous step.

As mentioned above, φ_1 can be set to any value and φ_2 can always be chosen in such a way that the angle ψ_{12} is preserved exactly. For the remaining angles φ_k

Algorithm 1 **Greedy MDS$_{polar}$**

```
$X = \{x_1, x_2, \ldots, x_n\}$
Let $\Psi_{n \times n}$ be a matrix with the pairwise angles $\psi_{ij}$ between all $(x_i, x_j)$
$\varphi_1 = 0$
for $k = 2$ to $n$ do
  $a_{ik} = 1$ for all $i = 1 \ldots k-1$
  for $i = 1$ to $k-1$ do
    $\varphi_k = \frac{\sum_{j=1}^{k-1} \varphi_j - \sum_{j=1}^{k-1} a_{jk}\psi_{jk}}{k-1}$        $e_k = \sum_{j=1}^{k-1} (\varphi_j - \varphi_k - a_{jk}\psi_{jk})^2$
    $t = \varphi_k$
    $a_{ik} = -1$
    $\varphi_k = \frac{\sum_{j=1}^{k-1} \varphi_j - \sum_{j=1}^{k-1} a_{jk}\psi_{jk}}{k-1}$        $f_k = \sum_{j=1}^{k-1} (\varphi_j - \varphi_k - a_{jk}\psi_{jk})^2$
    if $e_k < f_k$ then
      $a_{ik} = 1$
      $\varphi_k = t$
    end if
  end for
end for
```

no guaranty can be given that the greedy algorithm finds the optimal solution. Incremental adding of feature vectors can be achieved by simply extending the outer for-loop for another iteration for each new object. The angle φ_k will be computed analogously as for previous feature vectors.

3.4 Relative MDS$_{polar}$

As for conventional MDS, also for MDS$_{polar}$ different approaches regarding the objective function are feasible. The solution described above minimizes the absolute difference of pairwise angles of the original dataset and the transformed dataset. Large angles, which cause in tendency a large E may effect the solution in that way, that the transformation will represent vectors with small angles to others less correctly. Considering the relative error leads to

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} \left(\frac{\varphi_i - \varphi_k - a_{ik}\psi_{ik}}{\psi_{ik}} \right)^2 \tag{7}$$

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} \left(\frac{\varphi_i - \varphi_k - a_{ik}\psi_{ik}}{\psi_{ik}} \right) \frac{1}{\psi_{ik}}. \tag{8}$$

The greedy algorithm (1) can be applied only modifying the calculation specification for φ_k

$$\varphi_k = \frac{\sum_{i=1}^{k-1} \frac{\varphi_i}{\psi_{ik}^2} - \sum_{i=1}^{k-1} a_{ik} \frac{1}{\psi_{ik}}}{\sum_{i=1}^{k-1} \frac{1}{\psi_{ik}^2}}. \tag{9}$$

Because of the different objective functions the validity of solutions with the absolute MDS$_{polar}$ and the relative MDS$_{polar}$ can not be compared by means of E.

4 Weighted MDS$_{polar}$

In certain cases the objective when transforming data is to preserve relations of feature vectors of the original feature space in the target feature space. Thus, feature vectors that form a cluster should be represented as exact as possible in the target feature space, too. The transformation of feature vectors with a large distance to the respective feature vector can have a lower accuracy. An approach to achieve this goal is the introduction of weights w_{ik} to our objective function

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} w_{ik} (\varphi_i - \varphi_k - a_{ik}\psi_{ik})^2. \tag{10}$$

Determine the derivative leads to

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} w_{ik} (\varphi_i - \varphi_k - \psi_{ik}) \tag{11}$$

and solving for φ_k

$$\varphi_k = \frac{\sum_{i=1}^{k-1} w_{ik} (\varphi_i - a_{ik}\psi_{ik})}{\sum_{i=1}^{k-1} w_{ik}}. \tag{12}$$

Note that this is a generalization of relative MDS$_{polar}$. For relative MDS$_{polar}$, we simply choose the weights as $w_{ik} = 1/\psi_{ik}^2$.

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:

$$w_{ik} = w(l_i, l_k) = w(z). \tag{13}$$

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

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

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

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

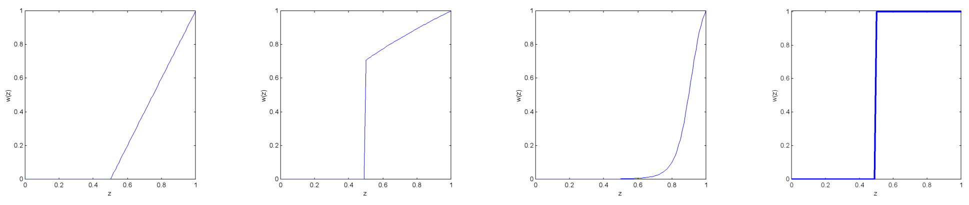


Fig. 1. Different Weighting Functions

might be useful if a certain threshold can be determined, beyond which difference in the relative distance between two feature vectors, the angle between them need not have any effect on the calculation of the respecting φ_k. To decrease the computational complexity, weights should be chosen in such a way, that for feature vectors with a certain (large) distance the respecting weights become zero. The following function describes a simple weighting function, which is the second function shown in Figure 1:

$$w(z_r) = \begin{cases} \sqrt{\left(\frac{z_r-\vartheta}{1-\vartheta}\right)} & \text{, if} \quad z_r \geq \vartheta \\ 0 & \text{, otherwise} \end{cases} \tag{14}$$
$$\text{where} \quad \vartheta \in [0,1].$$

With the threshold ϑ one can control indirectly the fraction of the data, that will be used to determine the respective angle φ_k. Thus, small values for ϑ lead to lots of weights w$\neq 0$ 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. When determining the weights for the calculation of φ_k it is sufficient to consider the feature vectors starting from index k. Weights will be calculated stepwise. With every step the weights become smaller until a weight becomes zero. Since the weighting function is decreasing, a further iteration would lead to zero, too. Thus, the calculation of weights stops at this point. 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 iterations for the calculation of the weights than setting a larger threshold. In this case, the projected vectors will be forced to a proper position already by a significantly large fraction of other feature vectors in the dataset. It might also be useful to reduce ϑ locally, when only few vectors satisfy the condition in equation (14).

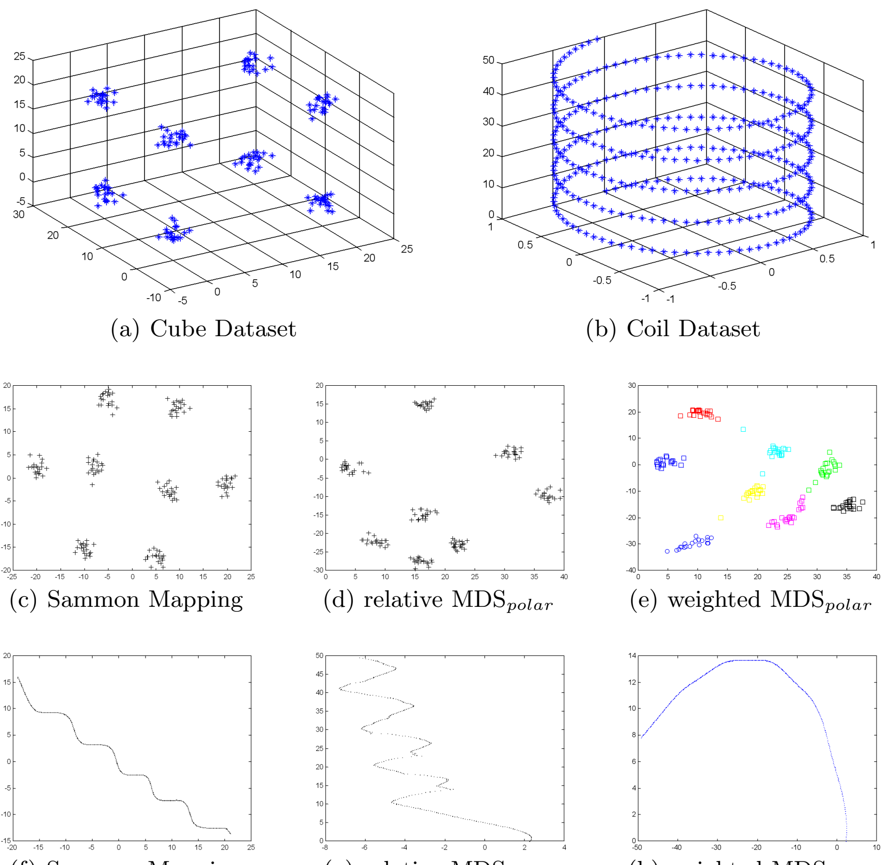
With a limitation of the number of weights $w > 0$ and a moderate ϑ at the same time, it can be achieved that the number of weights considered for the calculation of φ_k does not differ too much for different φ_k and limited computation time can be guarantied. Instead of considering the angles of all feature vectors with the greedy algorithm (1) it might be useful to consider only few feature vectors and calculating the exact solution of the sign problem. Using a weighting function enables the user of MDS$_{polar}$ to set a certain bin size which indicates the number of feature vectors that will be considered when calculating the desired φ_k. By means of this one can reduce the computation time and reinvest it in finding the exact solution of the sign problem. Thus, the upper bound for the complexity of our algorithm is due to sorting the data which is $O(n \cdot \log n)$. Solving the sign problem for a given maximum bin size b with the greedy strategy and using a certain number c of iterations, this accounts with $O(n \cdot b \cdot c)$ to the entire algorithm.

Evaluation of the transformation can be done by determining the average deviation from the original angles. In general this can be obtained by dividing E through the number of terms summed up. For the error function (4) one has to divide whith $\frac{n^2+n}{2}$. With this measured value one can compare different mappings even if they vary in the number of objects.

5 Results

Figure 2 shows some results of MDS$_{polar}$ in comparison with the Sammon Mapping. In favour of an easy verification of the results we applied MDS$_{polar}$ to some 3-dimensional datasets. The validity of the solution can be evaluated by visual inspection. The cube dataset (a) is about a synthetic dataset, where data points scatter around the corners of a 3-dimensional cube. Thus, the cube dataset contains eight well separated clusters. The dataset in (b) is comparable to a serpentine. As the figures (d) and (g) show, the transformations of MDS$_{polar}$ are similar to these of conventional MDS. Whereas MDS needs some thousand iterations until convergence, MDS$_{polar}$ finds an explicit solution after solving the system of equations. The transformations in figure (e) and (h) result from weighted MDS$_{polar}$ with weighting functions where at most twelve weights got values greater than zero. Thus, the transformation is based only on a relatively small number of angle comparisons. Therefore, locally these transformations are very accurate, but generally the loss of information is sometimes higher.

Since the value of φ_k is calculated from all preceding $\varphi_1 \dots \varphi_{k-1}$ according to equation (6) or equation (9) respectively, a solution with MDS$_{polar}$, either absolute or relative, depends to some degree on the order of the dataset. Our tests have shown, that in such cases only few feature vectors lead to higher errors, while others will not. Thus, not the complete transformation will be wrong, but only some feature vectors. Initialization is also a matter of fact of conventional MDS.


(a) Cube Dataset (b) Coil Dataset

(c) Sammon Mapping (d) relative MDS$_{polar}$ (e) weighted MDS$_{polar}$

(f) Sammon Mapping (g) relative MDS$_{polar}$ (h) weighted MDS$_{polar}$

Fig. 2. Different Transformations with MDS$_{polar}$

6 Conclusion

We presented a new approach for dimension reduction. MDS$_{polar}$ bases on the reduction of the error of the pairwise angle between feature vectors comparing angles of the original feature space with the angles in the transformed feature space. With MDS$_{polar}$ it is possible to add new feature vectors to the dataset and find a transformation for this feature vector without re-calculating the whole transformation. Our solution is explicit, which leads here to short computation time. Furthermore, we presented a greedy algorithm to get an approximation of the exact solution.

With weighted MDS$_{polar}$ we have introduced a weighting function with the objective to differentiate ones feature vector's importance to the approximation of the respecting φ_k. Non-similar feature vectors contribute less to the accuracy of the result than similar feature vectors. If such weighting functions are designed in such a way that a (large) fraction of the angles φ_i gets zero weight, then an exact solution of the sign problem can be found within moderate computation time. Our tests have shown that good solutions can be already found with 10 non-zero weights. Our examples approve that this approach is promising. Developing appropriate approximation schemes will be subject of future work. Furthermore, we plan to modify this technique to learn a function that maps feature vectors to the 2-dimensional feature space. New objects could be mapped even simpler to the plane.

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. In: Proceedings ACM SIGMOD International Conference on Management of Data, San Jose, CA (1995), 163–174.
4. Kruskal, J.B., Wish, M.: Multidimensional Scaling. SAGE Publications, Beverly Hills (1978).
5. Lowe, D., Tipping, M.E.: Feed-Forward Neural Networks Topographic Mapping for Exploratory Data Analysis. Neural Computing and Applications, 4, (1996), 83–95.
6. Morrison, A., Ross, G., Chalmers, M.: Fast Multidimensional Scaling through Sampling, Springs and Interpolation. Information Visualization (2003) 2, 68–77.
7. Williams, M., Munzner, T.: Steerable, Progressive Multidimensional Scaling. 10th IEEE Symposium on Information Visualization, Austin, TX (2004), 57–64