RC 20749 (02/28/97)
Computer Science/Mathematics

IBM Research Report

Exploratory Data Analysis in Large, Sparse Datasets

Ramesh Natarajan

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Exploratory Data Analysis in Large, Sparse Datasets

by

Ramesh Natarajan
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, N.Y. 10598.
e-mail: ramesh@watson.ibm.com

Abstract

Many applications of exploratory data analysis involve multivariate datasets that are large and high-dimensional, but quite sparse. Existing methods and computational algorithms are either expensive or inappropriate for these datasets. In this paper, we describe a modification of the Kohonen self-organizing maps algorithm for clustering and segmentation, whose storage and computational requirements are proportional to the data sparsity, rather than to the dimensions of the dataset. We also describe the use of a multidimensional scaling procedure that significantly improves the topological representation of the clusters obtained by the self-organizing maps algorithm. This methodology can be used in various applications including the analysis of retail shopping and credit card spending data, and text document indexing and classification.

1 Introduction

Existing methods and computational algorithms for exploratory data analysis (EDA) can be inappropriate and expensive for large, high-dimensional, sparse, multivariate data sets. Such data sets are frequently encountered in applications, for example:

1. A supermarket sales database where the attribute space is some categorization of the supermarket inventory, and each record contains the money spent in each category for an individual shopping-cart transaction.
2. An indexed collection of text files or documents, where the attribute space is the list of all words in the collection, and each document is characterized by its individual word counts.
3. A credit-card spending database where the attribute space is some categorization of the type of spending (i.e., entertainment, retail, gasoline and so on), and the records contain the spending data of different account holders.

If we denote the records in a data set by $\{\boldsymbol{X}_i\}_{i=1}^N$, where $X_i \in \mathbb{R}^M$ and N, M respectively denote the number of records and attributes, then each record corresponds to row in the data matrix $\boldsymbol{X} = \{X_{i,j}\}$, where $\boldsymbol{X} \in \mathbb{R}^{N\times M}$. In the examples given above, N and M can be potentially very large, but $\boldsymbol{X}$ is quite sparse with each record having only a few non-zero attributes (also the entries of $\boldsymbol{X}$ are all non-negative, so that the $\{\boldsymbol{X}_i\}_{i=1}^N$ lie in the positive orthant in $\mathbb{R}^M$).

As mentioned earlier, even the most elementary EDA methods can be inappropriate or expensive for these data sets. For example, the univariate statistics (such as means and standard deviations) along the marginals are skewed by the fact only a small fraction of the records may have a non-zero contribution to the given marginal. Even when the marginal statistics are suitably computed, standard transformations such as centering and scaling cannot be used since these transformations destroy the valuable sparsity property of the data matrix $\boldsymbol{X}$.

The use of standard methods on such data sets requires some preliminary reduction of the dimensionality and size of the data sets (without discarding the information content that is relevant to the objectives of the analysis). Since we are often interested in clustering or segmenting the records in the data set according to their characteristics, a useful dimensionality reduction technique is multidimensional scaling (MDS). Here, the data records are mapped by a nonlinear projection from $\mathbb{R}^M$ to $\mathbb{R}^d$ (for some $d \ll M$), so that the low-dimensional projection in $\mathbb{R}^d$ best preserves the metric relationships between the data records in $\mathbb{R}^M$ (see Kruskal and Wish [17] for an introductory discussion; also

Sammon [29], Borg and Lingoes [3]). If d is small and if the metric relationships are well-preserved, then the structure and variability of the resulting scatterplots in the projected MDS plot, which are easily visualized, can be used interpret the original high-dimensional data set. Fom a computational point of view, the MDS procedure takes the $N(N-1)/2$ distances in $\mathbb{R}^M$ as input and produces a map of N points in $\mathbb{R}^d$, by an optimization procedure involving Nd degrees of freedom. For large N, this optimization procedure is computationally expensive, and depending on the noise in the data, the difficulties due to the presence of suboptimal local minima solutions are also increased.

One approach towards a tractable MDS procedure is to first summarize the data set in terms of a smaller spanning set of D cluster prototypes, with each prototype representing a some subset of the original N records (similar approaches have been taken by Pykett [24] and Flexer [7]). In some data sets, these prototypes might represent clusters with specific distributions and inter-cluster separations (for example, they may be the individual sample means of mixture of Gaussians). However, for the large, heterogenous data sets arising from practical applications, it is more realistic to consider these prototypes as representing a quantization of the data (see Fukunaga [9], Chapter 11) that preserves the overall structure and variation in the data, while removing noise and small-scale variability.

Many standard quantization algorithms for determining these prototypes are prohibitively expensive for large data sets. These include, for example, all clustering algorithms that in their standard implementation require the similarity or distance matrix, because of the $O(N^2M)$ cost of generating and $O(N^2)$ cost of storing this matrix. One class of practical quantization algorithms is that based on the adaptive K-means methods and their generalizations (MacQueen [21], Kohonen [16], Marroquin and Girosi [22]). Here, starting from some initial guess, the prototypes are iteratively adjusted so as to minimize the RMS deviation of the records in the data set from their corresponding prototypes. The advantage of this method is that the prototypes can be computed in a small number of passes over the sparse data (when implemented as shown below in Section (2)). The adaptive K-means approach has the limitation that the number of prototypes is specified as an input to the algorithm, rather than being extracted from the "natural" clustering in the data. However, this is less of an issue when the objective is to summarize the data rather than to explicitly identify any clusters with specific distributional assumptions.

The use and interpretation of clustering methods must consider the issues of attribute scaling, record normalization, and the choice of the similarity metric. For example, many data sets contain attributes whose raw measurements may be on widely differing or even incompatible scales, and various transformations and scalings must be applied to ensure that compatible units of measurement are used for each marginal attribute (the use of various "automatic" scaling procedures in classical cluster analysis studies is discussed in Gnanadesikan, Kettenring and Tsao [11]). The role of record normalization and the choice of similarity metrics on the clustering results can be illustrated by following example with 3 records, $\boldsymbol{X}_1 = [0, 5, 0]$, $\boldsymbol{X}_2 = [1, 1, 0]$, and $\boldsymbol{X}_3 = [0, 0, 3]$. Then, without record normalization, the euclidean-distance similarity metric yields $||\boldsymbol{X}_1 - \boldsymbol{X}_2||_2 = \sqrt{17} > ||\boldsymbol{X}_3 - \boldsymbol{X}_2||_2 = \sqrt{11}$, with the paradoxical result that $\boldsymbol{X}_2$ and $\boldsymbol{X}_3$ are more similar even though they have no non-zero components in common. The emphasis on the pattern of the non-zero components can be recovered by using an angle-based distance metric for vectors in the positive orthant. This is metrically equivalent to unit normalizing the records, which maps each record to the unit hypersphere, and then using the euclidean metric. However, this transformation removes any overall scale information in the data, and this may be equally undesirable depending on the application. An even more drastic transformation is to rescale all non-zeros in the data matrix to unity, so that each record is mapped to the corners of the unit hypercube in the positive orthant (so that all attributes now only take binary categorical values). This particular transformation emphasizes the sparsity pattern, but removes both overall and relative scale information. In general, the scaling and transformation of the data provides analytical flexibility that can be used to modify and emphasize the pattern, relative scale and overall scale information, for a better understanding of the data.

We can summarize the methodology described in this paper as follows. First, the marginal univariate distributions are examined, and based on this, a preliminary scaling and normalization of the data is carried out (the details are quite specific to the application, and to the nature and quality of the data). Second, the self-organizing maps (SOM) algorithm is used as described below to summarize the data in terms of a small number of cluster prototypes, and at the same time to construct a preliminary low-dimensional representation of the inter-cluster relationships. Third and finally, the quality and accuracy of these low-dimensional representations is improved by an MDS procedure, for the final presentation and interpretation of the results. The details of the second and third steps above are described in Section (2) below. Section (3) consider the application of this methodology in understanding the internal structure and relationships in a set of text documents.

2 Computational Algorithms

2.1 Self-Organizing Maps

The self-organizing maps (SOM) algorithm originates in the neural network literature (Kohonen [16]), and is often used as a heuristic, non-parametric, K-means type, clustering procedure that also provides a low-dimensional "topology-preserving" representation of the ensuing clusters. The algorithm, as described below, computes certain "weight vectors" $\{\boldsymbol{w}_i\}_{i=1}^{D}$, $\boldsymbol{w}_i \in \mathbb{R}^M$, which are the locations of the D cluster centroids, along with a one-to-one mapping of these weight vectors onto a low-dimensional configuration $\mathcal{L}$ consisting of D nodes whose point-coordinates are given by $\{\boldsymbol{l}_i\}_{i=1}^{D}$, $\boldsymbol{l}_i \in \mathbb{R}^d$ with $d \ll M$. In the neural network literature, $\mathcal{L}$ is typically a regular 2-D lattice, but in our description the structure and dimensionality of $\mathcal{L}$ is not necessarily so constrained. In summary, the SOM algorithm computes the weight-vectors $\{\boldsymbol{w}_i\}_{i=1}^{D}$ and maps these onto the nodes $\{\boldsymbol{l}_i\}_{i=1}^{D}$ such that the $\boldsymbol{w}_i$ that are close to each other in terms of their euclidean distance in $\mathbb{R}^M$ are mapped to nearby nodes on $\mathcal{L}$. The procedure that is used is as follows.

<u>Algorithm SOM:</u>

1. Start with some initial weight vectors $\{\boldsymbol{w}_j\}_{j=1}^{D}$.
2. For each pass over the input data $p = 1, N_P$

   For each input record from $\{\boldsymbol{X}_i\}_{i=1}^{N}$

   (a) Compute distances d_j to each weight vector
   $$d_j^2 = ||\boldsymbol{X}_i - \boldsymbol{w}_j||^2, \quad j = 1, \ldots, D. \tag{1}$$
   (b) Find the closest weight vector to the input record
   $$k_i = \arg\min_j d_j^2. \tag{2}$$
   (c) Update weight vectors at all nodes $j \in \mathcal{L}$ that are in some neighborhood of node k_i
   $$\boldsymbol{w}_j \longleftarrow \boldsymbol{w}_j + \alpha_p(\widehat{jk_i})(\boldsymbol{X}_i - \boldsymbol{w}_j). \tag{3}$$

   Endfor

   Endfor

In Step (1), the weight vectors can be initialized with randomly selected input data records. In Step (2), the weight vectors are incrementally modified to their final values in N_P passes over the the input data (typically we take N_P in the range 100 to 500). In each such pass, each input record is compared to the current weight vectors and the nearest weight vector denoted by k_i, along with the weight vectors of neighboring nodes j of k_i, are incrementally updated as in (3). The function $\alpha_p(\widehat{jk_i})$ appearing here can be written as

$$\alpha_p(\widehat{jk_i}) = \epsilon_p h_p(\widehat{jk_i}), \tag{4}$$

where ϵ_p is a learning rate parameter (whose value is monotonically decreasing, typically we take 0.98 in the first pass and reduce to about 0.2 in the final pass), and h_p is a "neighborhood" function centered on the winning node k_i, whose argument is the distance $\widehat{jk_i}$ between the node j and k_i. A typical choice for the neighborhood function is the Gaussian

$$h_p(\widehat{jk_i}) = e^{-||\boldsymbol{l}_j - \boldsymbol{l}_{k_i}||^2/2\sigma_p^2}, \tag{5}$$

and the value of the decay width σ_p in (5) is reduced monotonically with each pass over the input data (typically this is reduced starting from about half the diameter of $\mathcal{L}$ in the first pass, to a small value much less than the minimum node separation on $\mathcal{L}$ in the last pass).

This basic SOM algorithm and its many variants have been used in numerous applications (see Kohonen [16] and reference therein). We describe a reformulation of the basic SOM algorithm for sparse input data sets, termed SOM-SD (Self-Organizing Maps for Sparse data), which to our knowledge, has not previously appeared in the literature. For sparse data sets, this SOM-SD reformulation often provides significant computation and storage savings over the equivalent in-core implementation of the original SOM algorithm (in many practical data sets, we have seen storage and computation savings of a factor of 10 or more).

This SOM-SD reformulation only requires the the non-zero entries of $\boldsymbol{X}$ which are assumed to be stored in the standard row-compressed sparse notation (Duff, Erisman and Reid [6]). In addition, it uses 2 scalars associated with each node weight vector, denoted $\{\beta_j\}_{j=1}^D$ and $\{n_j\}_{j=1}^D$ respectively. It also uses an auxiliary "normalized" set of weight vectors $\{\boldsymbol{z}_j\}_{j=1}^D$ (which are introduced for clarity of exposition; in the program implementation, these are overwritten by the actual weight vectors). The idea is that even though the weight vectors $\{\boldsymbol{w}_j\}_{j=1}^D$ are not necessarily sparse, by defining $\boldsymbol{w}_j = \beta_j \boldsymbol{z}_j$ and $n_j = ||\boldsymbol{z}_j||^2$, the values of $\{\beta_j, n_j, \boldsymbol{z}_j\}_{j=1}^D$ can be updated for each input record $\boldsymbol{X}_i$ at a cost proportional to the number of non-zeros in $\boldsymbol{X}_i$. For example, to compute the distance from $\boldsymbol{X}_i$ to the the weight vector $\boldsymbol{w}_j$, we have

$$d_j^2 = ||\boldsymbol{X}_i - \boldsymbol{w}_j||^2 = \beta_j^2 n_j - (2\beta_j \boldsymbol{z}_j - \boldsymbol{X}_i)^T \boldsymbol{X}_i. \tag{6}$$

and if β_j and n_j are known, the remaining inner-product can be evaluated at a cost proportional to number of non-zeros in $\boldsymbol{X}_i$. Furthermore, if k_i is the index of the closest weight vector, then the new values for β_j and $\boldsymbol{z}_j$ in the update step are given by

$$\beta_j^{new}\boldsymbol{z}_j^{new} = (1-\alpha_p(\widehat{jk_i}))\beta_j\boldsymbol{z}_j + \alpha_p(\widehat{jk_i})\boldsymbol{X}_i, \tag{7}$$

Here, by first setting $\beta_j^{new} = (1-\alpha_p(\widehat{jk_i}))\beta_j$, we have

$$\boldsymbol{z}_j^{new} = \boldsymbol{z}_j + \frac{\alpha_p(\widehat{jk_i})}{\beta_j^{new}}\boldsymbol{X}_i. \tag{8}$$

and the latter update can be carried out at a cost proportional to the number of non-zeros in $\boldsymbol{X}_i$. At the same time, the new value for n_j can also be obtained from

$$n_j^{new} = ||\boldsymbol{z}_j^{new}||^2 = n_j + (2\boldsymbol{z}_j + \frac{\alpha_p(\widehat{jk_i})}{\beta_j}\boldsymbol{X}_i)\frac{\alpha_p(\widehat{jk_i})}{\beta_j}\boldsymbol{X}_i, \tag{9}$$

in which the required inner-product can be evaluated at a cost proportional to the number of non-zeros in $\boldsymbol{X}_i$. The use of the superscript *new* in (7)-(9) is only for clarity of exposition, and in the program implementation the new values overwrite the existing values. The details of the procedure are as follows.

<u>Algorithm SOM-SD:</u>

1. Start with some initial weight vectors $\{\boldsymbol{w}_j\}_{j=1}^D$.
2. For each pass over the input data $p = 1, N_p$
   (a) Set $\beta_j = 1, n_j = ||\boldsymbol{w}_j||^2, j = 1, \ldots, D$.
   (b) For each input record from $\{\boldsymbol{X}_i\}_{i=1}^N$
      i. Compute distances d_j to each weight vector

$$d_j = \beta_j^2 n_j - (2\beta_j\boldsymbol{z}_j - \boldsymbol{X}_i)^T\boldsymbol{X}_i. \tag{10}$$

      ii. Find the node whose weight vector is closest to the input record

$$k_i = \arg\min_j d_j. \tag{11}$$

      iii. Update $\{\beta_j, n_j, \boldsymbol{z}_j\}$ at all nodes j on $\mathcal{L}$ that are in some neighborhood of node k_i

$$\begin{aligned}\beta_j &\longleftarrow (1-\alpha_p(\widehat{jk_i}))\beta_j,\\ n_j &\longleftarrow n_j + (2\boldsymbol{z}_j + \frac{\alpha_p(\widehat{jk_i})}{\beta_j}\boldsymbol{X}_i)\frac{\alpha_p(\widehat{jk_i})}{\beta_j}\boldsymbol{X}_i,\\ \boldsymbol{z}_j &\longleftarrow \boldsymbol{z}_j + \frac{\alpha_p(\widehat{jk_i})}{\beta_j}\boldsymbol{X}_i,\end{aligned} \tag{12}$$

   Endfor
   (c) Renormalize to obtain the weight vectors $\boldsymbol{w}_j = \beta_j\boldsymbol{z}_j, j = 1, \ldots, D$.

Endfor

Since $0 < \alpha_p(\widehat{jk_i}) \leq 1$, the values of β_j computed in this algorithm form a decreasing sequence, and in order to avoid division by small quantities, whenever (12) indicates that the new value of a certain β_j will be below a certain threshold (which we typically set to 0.01), then for that value of j, the following alternative code sequence to (12) is executed.

$$\begin{aligned} \boldsymbol{z}_j &\longleftarrow \beta_j \boldsymbol{z}_j, \\ \beta_j &\longleftarrow 1, \\ \boldsymbol{z}_j &\longleftarrow (1 - \alpha_p(\widehat{jk_i}))\boldsymbol{z}_j + \alpha_p(\widehat{jk_i})\boldsymbol{X}_i, \\ n_j &\longleftarrow ||\boldsymbol{z}_j||^2. \end{aligned} \tag{13}$$

Although the evaluation of n_j and $\boldsymbol{z}_j$ in (13) is no longer determined by the sparsity of $\boldsymbol{X}_i$, this alternate sequence is executed so infrequently that its contribution to the overall running time of the algorithm is insignificant. Note that taking $\epsilon_p = 0.98$ in the first pass (see following the description of SOM above), and setting the threshold to 0.01 as above, ensures that this alternate code sequence is not triggered for the updates to each winning node k_i in this first pass .

The quality of the clustering results from the SOM/SOM-SD algorithm can be evaluated using the RMS error index

$$\mathcal{Q} = \sqrt{\frac{\sum_{i=1}^{N} ||\boldsymbol{X}_i - \boldsymbol{w}_{k_i}||^2}{N}}. \tag{14}$$

which indicates the mean dispersion of the input records about their cluster prototypes. This index, which implicitly depends on the number D of cluster prototypes, can be evaluated in an extra pass over the sparse input data. The quality of the topological mapping of the clusters on $\mathcal{L}$ can be evaluated and optimized by another index, as discussed in the next subsection.

2.2 Multidimensional Scaling

In practice, the heuristic SOM algorithm is limited by the fact that the point-configuration $\mathcal{L}$ in $I\!R^d$ is chosen in a somewhat arbitrary fashion (for example, as a 2-D regular lattice). Therefore, although the node proximities in $\mathcal{L}$ may in fact reflect the topological relationships between the corresponding weight vectors, the constraint of a regular lattice makes it incapable of accurately representing the actual metric relationships, making the results somewhat more difficult to interpret than a standard MDS plot. In other cases, the assumed dimensionality and structure for $\mathcal{L}$ in SOM/SOM-SD may be completely unsuitable for the data, so that even topological relationships are not accurately described (with nearby weight vectors being mapped to nodes that are far apart, and conversely distant weight vectors are mapped to nearby nodes on $\mathcal{L}$). These limitations of SOM have also been noted by others (Bezdek and Nikhil [5], Flexer [7]).

Nevertheless, the SOM results provide a summarization of the data, along with a good starting point for a subsequent MDS analysis in which the configuration $\mathcal{L}$ is adjusted to remove some of the limitations described above. We therefore let $\boldsymbol{W} \in I\!R^{D\times M}$ denote the matrix whose rows are the weight vectors $\{\boldsymbol{w}_j\}_{j=1}^D$, and $\boldsymbol{L} \in I\!R^{D\times d}$ denote the matrix whose rows are the associated point coordinates $\{\boldsymbol{l}_i\}_{i=1}^D$ of $\mathcal{L}$. In order to remove the indeterminacy associated with the translation and dilation of the coordinates of $\mathcal{L}$ from the MDS objective function defined below, we introduce the matrix $\boldsymbol{L}' \in I\!R^{D\times d}$ whose rows are given by the point coordinates $\{\boldsymbol{l}'_i\}_{i=1}^D$ of the equivalent normalized configuration $\mathcal{L}'$, where

$$\boldsymbol{l}'_j = \frac{\boldsymbol{l}_j - \boldsymbol{\mu}}{[tr(\boldsymbol{S})]^{1/2}}, \tag{15}$$

where

$$\boldsymbol{\mu} = \frac{1}{D}\sum_{j=1}^{D} \boldsymbol{l}_j, \quad \boldsymbol{S} = \frac{1}{D-1}\sum_{j=1}^{D}(\boldsymbol{l}_j - \boldsymbol{\mu})(\boldsymbol{l}_j - \boldsymbol{\mu})^T, \tag{16}$$

In (16) $\boldsymbol{\mu}$ and $\boldsymbol{S}$ respectively denote the mean and covariance matrix of $\mathcal{L}$. The choice of the trace of the covariance matrix $tr(\boldsymbol{S})$ (which is sum of the marginal variances) as the scaling parameter ensures that any two configurations that only differ by an orthogonal transformation about their centroids have the same scaling factor, and therefore lead to the same value for the MDS objective function in (18) below. This means, however, that the optimization procedure can still produce two converged solutions that are identical upto an orthogonal transformation. Rather than remove this indeterminacy (either by redefining the objective function, or by modifying the optimization procedure) which would considerably complicate the computational details, our approach is to simply compute numerous converged solutions by the relatively straightforward optimization procedure described below, and eliminate the solutions that are orthogonally-transformed "copies" of existing solutions in a postprocessing step.

If we now denote the distance matrices in $I\!R^M$ and $I\!R^d$ by $\boldsymbol{P}$ and $\boldsymbol{Q}$ respectively, whose entries are given by

$$p_{ij} = ||\boldsymbol{w}_i - \boldsymbol{w}_j||, \quad q_{ij} = ||\boldsymbol{l}'_i - \boldsymbol{l}'_j||, \tag{17}$$

then the following goodness-of-fit or "stress" index between the two distance matrices can be defined

$$\mathcal{I}(f, \boldsymbol{L}) = \frac{1}{D(D-1)} \sum_{i=1}^{D} \sum_{i<j}^{D} [f(p_{ij}) - q_{ij}]^2, \tag{18}$$

where the function f is a non-negative, monotonic function with $f(0) = 0$. The MDS problem can then be formulated as that of finding

$$\{f^*, \boldsymbol{L}^*\} = \arg\min_{f,\boldsymbol{L}} I(f, \boldsymbol{L}). \tag{19}$$

The function f may be regarded as a "model" for the $\{p_{ij}, q_{ij}\}$ scatterplot. A linear model for f is most intuitive and easy to interpret, since it implies a simple scaling between $\boldsymbol{P}$ and $\boldsymbol{Q}$. However, a more general non-linear model for f might provide a better fit for the $\{p_{ij}, q_{ij}\}$ scatterplot data. In particular, it may lead to an equally good fit in a MDS plot of lower dimension (leading to a more compact description, with a more useful interpretation of the data, see Section (3)).

Rather than make specific parametric assumptions for f, we have used polynomial models (subject to the monotonicity constraint)

$$f(x) = \sum_{j=1}^{n} a_j x^j, \quad \text{subject to } \sum_{j=1}^{n} j a_j x^{j-1} \geq 0 \text{ for } x \geq 0. \tag{20}$$

The coupling of f and $\boldsymbol{L}$ in the objective function in (19) along with the assumed model for f in (20) leads to a difficult constrained, nonlinear optimization problem. Our approach is to decouple the difficulties due to the constraints and the nonlinearity by the following iterative procedure in which f and $\boldsymbol{L}$ are updated separately. The resulting implementation is simple to program, and makes use of standard and reliable subroutines for each computational step, as shown below.

<u>Algorithm MDS:</u>

1. Perform a SOM/SOM-SD clustering analysis for some arbitrary configuration $\mathcal{L}$ with a node distance matrix $\boldsymbol{L}$. Obtain the distance matrix $\boldsymbol{W}$ from the computed weight vectors. Use $\boldsymbol{L}$ as an initial guess $\boldsymbol{L}^{(0)}$ to the MDS configuration in $\mathbb{R}^d$.
2. For i = 1, ..., until convergence, do
   (a) Obtain the distance matrix $\boldsymbol{Q}^{(i-1)}$ for the normalized configuration $\boldsymbol{L}'^{(i-1)}$.

(b) Keeping $\boldsymbol{L}^{(i-1)}$ fixed, solve a inequality-constrained, least-squares problem to obtain $f^{(i)}$ corresponding to the model in (20) from

$$f^{(i)} = \arg\min_{f} \mathcal{I}(f, \boldsymbol{L}^{(i-1)}). \tag{21}$$

(c) Keeping $f^{(i)}$ fixed, take one or more steps of a nonlinear optimization procedure to obtain a new MDS configuration $\boldsymbol{L}^{(i)}$ such that (20)

$$\mathcal{I}(f^{(i)}, \boldsymbol{L}^{(i)}) \leq \mathcal{I}(f^{(i)}, \boldsymbol{L}^{(i-1)}). \tag{22}$$

(d) If unconverged, go to Step (2a).

Similar approaches to this optimization problem have been previously used in multidimensional scaling (Kruskal and Wish [17], Borg and Lingoes [3]), and for the unconstrained case, Ruhe and Wedin [27] survey a number of algorithms for separable nonlinear least-squares problems, which include procedures that decouple linear and nonlinear variables in the objective function (similar to our procedure here). From (21) and (22), we note that

$$\mathcal{I}(f^{(i)}, \boldsymbol{L}^{(i)}) \leq \mathcal{I}(f^{(i)}, \boldsymbol{L}^{(i-1)}) \leq \mathcal{I}(f^{(i-1)}, \boldsymbol{L}^{(i-1)}), \tag{23}$$

so that the overall optimization procedure leads to a non-increasing sequence of values for the objective function. The use of a globally-convergent nonlinear optimization procedure in Step (2c), along with the uniqueness of the solution to the linearly-constrained least squares problem in Step (2b), ensures that the equality in (23) is achieved only at a converged local minimum of the objective function. The required uniqueness of the linearly-constrained least squares problem in Step (2b) follows from the strict convexity of the objective function on the convex constraint set (the set of coefficients of all monotonic polynomials of a given order).

In typical usage, the initial SOM/SOM-SD procedure is carried out with several different choices for the point-configuration $\mathcal{L}$, since each choice possibly leads to a different set of weight vectors and different values for RMS error $\mathcal{Q}$ in (14). In addition, even when the computed weight vectors are identical or differ only slightly, each configuration provides a reasonable starting point for obtaining different local minima solutions from the MDS optimization procedure. From these different results, the clustering and MDS configuration that have low values for both the $\mathcal{Q}$ and $\mathcal{I}$ indices are selected for final interpretation.

The solution of (21) using the model (20) is carried out using one of the procedures suggested by Hawkins [14]. The analytic constraints on the derivative are discretized at n_c points for x in the interval $[0, 1.5 \max_{i,j}(p_{ij})]$. The appropriate value for n_c will depend on the degree n of the fitting polynomial, and for the relatively smooth low-order polynomial fits that we have used, a value of 20 to 30 equi-spaced points in this range was generally adequate. The resulting linear-inequality constrained, least-squares problem for the parameters $\{a_j\}_{j=1}^{n}$ is solved using the LSEI code of Hanson and Haskell [12].

The optimization procedure in Step (2c) is carried out using an unconstrained nonlinear conjugate gradient solver, for which we have used the implementation of Shanno and Phua [30] (modified to allow the gradient evaluations to be passed by a reverse communication interface). This requires the analytical derivatives of the objective function (22), which can be obtained after some algebra as

$$\nabla_{\boldsymbol{l}_k}\mathcal{I} = \frac{2}{D(D-1)[tr(\boldsymbol{S})]^{1/2}} \sum_{i=1}^{D} \left[\{f(p_{ik}) - q_{ik}\} \frac{q_{ik}}{|q_{ik}|} + \frac{\boldsymbol{l}'_k}{(n-1)} \sum_{i<j}^{D} \{f(p_{ij}) - q_{ij}\} |q_{ij}| \right]. \tag{24}$$

The correctness of the analytical derivatives for the optimization procedure is checked by comparing to the equivalent finite-difference estimate (computed in the program as suggested in [2]).

The converged result $\boldsymbol{L}$ from the MDS algorithm is normalized to $\boldsymbol{L}'$ as in (15), and then finally rotated to a principal-axes coordinate system (using the SVD algorithm implemented in the DGESVD routine from LAPACK [1]). This principal axes transformation is useful in identifying the directions of the primary variation in the data. Furthermore, it provides a canonical basis for comparing two different solutions from the MDS procedure. In particular, two solutions that differ by an orthogonal transformation can often be easily identified in the MDS plots, although a more careful quantitative comparison of this nature will require the use of the Orthogonal Procrustes procedure (described in Borg and Lingoes [3], Golub and Van Loan [10]).

Finally, we note that the formulation described in this subsection differs in many details from comparable implementations of metric MDS procedures (such as the KYST program described in Kruskal, Young and Seery [18]). For example, our MDS objective function is based directly on using a reference configuration in which translation and scale effects are removed in a rotationally-invariant fashion, so that even intermediate solutions can be renormalized and rotated without affecting the value of the objective function or the convergence of the optimization procedure. Other improvements include the use of a least-squares polynomial fitting procedure that explicitly enforces a monotonicity constraint (although this constraint is often satisfied even when it is not explicitly enforced), as well as a more robust optimization procedure for the MDS objective function.

3 Results

In this section, we illustrate the use of the methodology in a practical application where the sparse nature of the data matrix must be taken into account (this example also illustrates the important role of data scaling and normalization in interpreting the results).

3.1 Document Clustering

Clustering methods have been used for classification, search and retrieval of text documents (see the exhaustive reviews in Willett [31] and Rasmussen [26]; Lin *et al* [19] and Merkl [23] have considered the use of self-organizing maps in this context). Document clustering has been use for information visualization in digital libraries and WWW document collections (Rao *et al* [25], Lagus *et al* [20]). As mentioned in Section (1), these applications typically lead to sparse feature vectors, and Berry, Dumais and O'Brien [4] have recently reviewed the role of sparse-matrix principal component analysis for this class of problems.

The data set used in this example was generated using 210 technical abstracts from ICASE, NASA Langley published in the period 1993 to 1996 (obtained from their World Wide Web site). This entire collection was parsed into its constituent words, and the Porter suffix stripping algorithm (Frakes [8]) was used to generate a set of 3065 unique stemmed terms for the corpus dictionary. Each document was characterized by a feature vector of the individual frequencies from the corpus dictionary appearing in the document, leading to a data matrix that had less than 3% nonzero entries. Since the abstracts were of roughly the same size, no row normalization was required to compensate for the document length variability (Salton and Buckley [28]).

If μ_{ij} denotes the frequency of the term j in document i, and $\mu_j = \sum_{i=1}^{N} \mu_{ij}$ the aggregate frequency over the corpus, then the following entropy-based measure s_j can be defined for term j

$$s_j = -(\log N)^{-1} \sum_N \frac{\mu_{ij}}{\mu_j} \log \frac{\mu_{ij}}{\mu_j}. \tag{25}$$

This takes values in the range 0 (when all the occurrences of the term are in a single document) to 1 (when the occurrences are uniformly distributed). A histogram of the s_j values for the terms in this data set is shown in Figure (1). A few terms like "abstract" and "NASA" appear in virtually all abstracts and have $s_j \longrightarrow 1$, while a more substantial fraction of the terms appear idiosyncratically in just a few abstracts and have $s_j \longrightarrow 0$. In order to damp the contributions of both ends of this distribution in the clustering

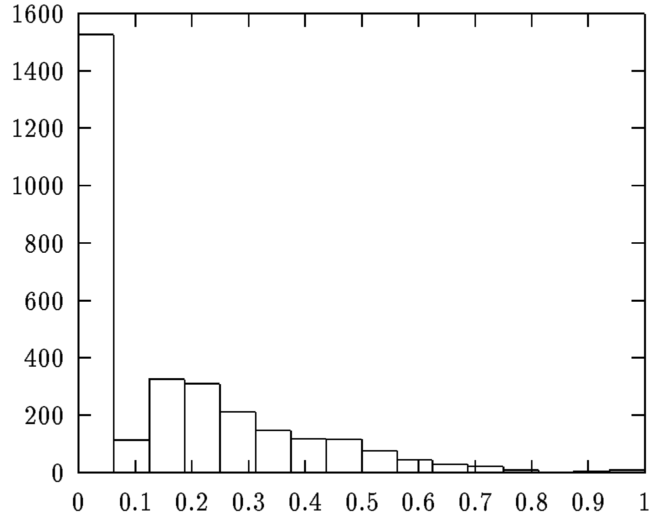

Figure 1: Histograms of the entropy-based term measure s_j for Example II

analysis, a heuristic term weighting factor $s_j(1 - s_j)$ was applied to each column j of the data matrix.

For the sake of comparison, the principal components of this transformed data matrix were computed (this problem was small enough that the dense-matrix DGESVD routine in LAPACK [1] could be used). The results indicated that 40, 88 and 139 leading PC's were required for respectively capturing 50%, 75% and 90% of the variation in the data. The number of PC's required to even capture 50% of the variation is fairly large, and in addition the corresponding "scree" plot (Jolliffe [15]) shows no indication of a clean separation between "signal" and "noise" subspaces in principal component variables. For larger data sets, where sparse iterative methods must invariably be used, the accurate computation of such a relatively large fraction of the PC's can be computationally prohibitive.

We now briefly summarize some of the results for the SOMS/MDS methodology. To fix ideas, $D = 10$ clusters were used in the SOMS procedure, and a cubic model $n = 3$ was used for f in the MDS procedure (increasing D and n did not change the results in a way that would impact the discussion below). Several randomly-generated initial configurations were used in 1-D, 2-D and 3-D studies, and in each case specific solutions with small values for both $\mathcal{Q}$ and $\mathcal{I}$ were selected for interpretation. The selected solutions had in the 1-D case ($\mathcal{Q} = 1.485, \mathcal{I} = 0.093$), in the 2-D case ($\mathcal{Q} = 1.47, \mathcal{I} = 0.037$), and in the 3-D case ($\mathcal{Q} = 1.468, \mathcal{I} = 0.012$). The corresponding MDS scatter-plots are shown in

Figure (2).

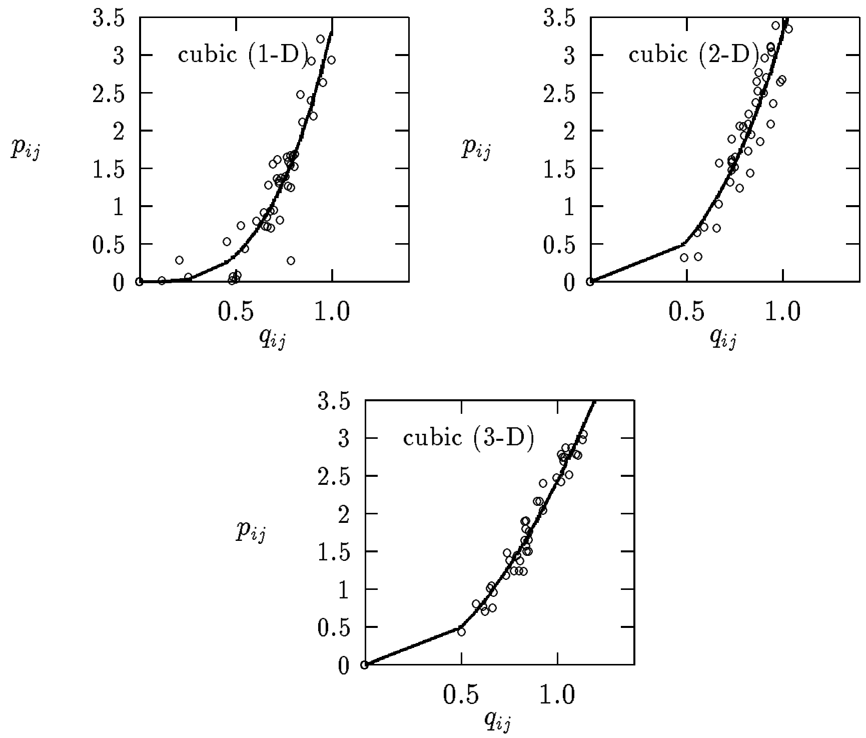

Figure 2: MDS scatter-plots for 1-D, 2-D and 3-D with $D = 10, n = 3$ for Example II

The 2-D MDS plot had 66% of the variance in the first dimension, while the 3-D MDS plot had 55% in the first and 85% in the first two dimensions. The 2-D results can be seen to give a reasonable description, and the corresponding MDS plot is shown in Figure (3), along with a description of each cluster's characteristics in Table (1). In this description, the characteristic terms for each cluster are those with the largest loadings in the corresponding weight vector. It is interesting to see the appearance of clusters in Figure (1) and Table (1) that reflect the broad areas of research at ICASE (such as experimental, theoretical and computational fluid dynamics, numerical methods, numerical optimization, parallel computing etc.). The primary variation in the MDS plot is along the horizontal axis and the most dissimilar clusters are 10 (which contains abstracts primarily in boundary-layer theory and stability from a theoretical fluid dynamics perspective) and 5 and 6 (which contains abstracts primarily on parallel computing with a computer science perspective). Another axis of variation is tilted from the vertical with cluster 1 (primarily

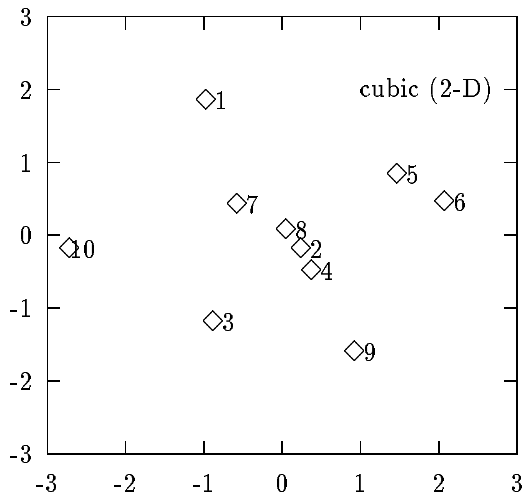

Figure 3: 2-D MDS plot for the SOMS clusters for Example II

turbulence theory) to cluster 9 (numerical methods including multigrid and unstructure CFD computations). Other notable clusters are 3 (vortex methods and vortical flows) and 4 (numerical optimization).

We make a specific note of cluster 8 in the center of the plot which contains many apparently unrelated abstracts, that individually have a fairly unique and distinctive content relative to the rest of the collection (this included abstracts on topics such as wavelets, discret-event simulation, combustion etc.). These abstracts, because of their unique individual content, tended to have many terms with ($s_j \longrightarrow 0$), along of course with the terms common to all abstracts with ($s_j \longrightarrow 1$). Therefore, the heuristic weighting scheme tends to damp the feature vectors of these abstracts close to the origin in the positive orthant. This is a nice feature of the attribute weighting scheme, which by mapping all "outlier" abstracts into this one cluster, leads to a more "robust" clustering methodology for identifying the clusters in the major part of the data.

Although the data set for this particular example was studied from an exploratory point of view, many variations and applications of this methodology can be envisioned. For example, one might examine the relationships between individual authors based on the content of the abstracts that they author or co-author. Berry, Dumais and O'Brien [4] have described an application of principal components analysis for content-based matching of technical conference abstracts and potential reviewers in the relevant areas of competency.

No.	No. of abstracts	most significant word-stems (in decreasing order)
1	18	*turbul, reynold, stress, model, closur, deriv*
2	27	*algorithm, method, problem, consid, analysi, converg*
3	13	*vortex, vortic, flow, numer, structur, interact*
4	22	*system, problem, optim, present, approxim, mail, exampl*
5	20	*parallel, perform, process*
6	19	*parallel, commun, perform, support, system, data*
7	22	*effect, studi, numer, turbul, field, layer, interact, acoust*
8	37	*condition, equat, function, not, boundari, given, show*
9	18	*unstructur, mesh, mavripli, grid, multigrid, techniqu*
10	14	*layer, stabil, boundari, number, disturb, investig, simul*

Table 1: Cluster characteristics in the 2-D MDS plot for Example II

4 Concluding Remarks

An efficient variant of the self-organizing maps (SOM) algorithm is described for computing cluster prototypes in large, high-dimensional, sparse data sets. This algorithm provides a low-dimensional, topological representation of the resulting cluster prototypes, which is used as an effective starting point for a multidimensional scaling (MDS) analysis to obtain a more accurate representation reflecting actual inter-cluster metric relationships. Although SOM and MDS have often been used as independent data analysis procedures in numerous applications (in the neural network and statistics literature respectively), the combination of the two methods leads to a computationally economical procedure for large, sparse data sets, with results that are easier to interpret and digest than would be possible using either procedure alone.

We contrast this approach with the common procedure for small or moderate data sets, in which a principal components or MDS procedure is first used to reduce the dataset dimensionality, and graphical or analytical methods are then used to identify clusters and other patterns in the low-dimensional transformed dataset. For larger data sets, this approach is computationally expensive, especially when as is often the case, a small number of PC's or a low-dimensional MDS plot does not provide a sufficiently accurate representatio of the data. In that case, our suggested alternative is to summarize and reduce the data set content using the SOM algorithm, prior to the application of an MDS analysis.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users Guide*, Second Edition, SIAM, Philadelphia (1995).

[2] V. Averbukh, S. Figueroa and T. Schlick, *ALGORITHM 566: HESFCN - A FORTRAN Package of Hessian Subroutines for Testing Nonlinear Optimization Software*, ACM Transactions on Mathematical Software, 20 (1994), pp. 282-285.

[3] I. Borg and J. C. Lingoes, *Multidimensional Similarity Structure Analysis*, Springer (1987).

[4] M. W. Berry, S. T. Dumais and G. W. O'Brien, *Using Linear Algebra for Intelligent Information Retrieval*, SIAM Review, 37 (1995), pp. 573-595.

[5] J. C. Bezdek and R. P. Nikhil, *An Index of Topological Preservation for Feature Extraction*, Pattern Recognition 28, (1995), pp. 381-391.

[6] I. S. Duff, A. M. Erisman and J. K. Reid , *Direct Methods for Sparse Matrices* Oxford (1986).

[7] A. Flexer, *Limitations of Self-Organizing Maps for Vector Quantization and Multidimensional Scaling*, to appear in Advances in Neural Information Processing Systems (eds., M. C. Mozer, M. I. Jordan and T. Petsche) 9 (1997).

[8] W. B. Frakes, *Stemming Algorithms*, in *Information Retrieval: Data Structures and Algorithms*, (eds. W. B. Frakes and R. Baeza-Yates), Prentice-Hall, (1992), pp.131-160.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Second Edition, Academic (1990).

[10] G. H. Golub and C. F. van Loan, *Matrix Computations*, Second Edition, Johns Hopkins University Press, Baltimore (1989).

[11] R. Gnanadesikan, J. R. Kettenring, and S. L. Tsao, *Weighting and Selection of Variables for Cluster Analysis*, 12 (1995), pp. 113-136.

[12] R. J. Hanson and K, H, Haskell, *ALGORITHM 587: Two Algorithms for the Linearly Constrained Least Squares Problem*, ACM Transactions on Mathematical Software, 8 (1982), pp. 323-333.

[13] J. A. Hartigan and M. A. Wong, *Algorithm AS 136: A K-means clustering algorithm*, Applied Statistics, 28 (1979), pp. 100-104.

[14] D. M. Hawkins, *Fitting Monotonic Polynomials to Data*, Computational Statistics, 9 (1994), pp. 223-247.

[15] I. T. Jolliffe, *Principal Components Analysis*, Springer (1986).

[16] T. Kohonen, *The Self-Organizing Map* Proceedings of the IEEE, 78 (1990), pp. 1464-1480.

[17] J. Kruskal and M. Wish, *Multidimensional Scaling*, (Quantitative Approaches to Social Sciences No. 14), Sage Publishers, Beverly Hills CA (1978).

[18] J. B. Kruskal, F. W. Young and J. B. Seery, *How to Use KYST: A Very Flexible Program To Do Multidimensional Scaling and Unfolding*, available from `http://www.netlib.org/mds/kyst2a_manual.ps`.

[19] S. Lin, D. Soergel and G. Marchionini, *A Self-Organizing Map for Information Retrieval*, in *Proc. Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Chicago, Illinois (1991), pp. 337-346.

[20] K. Lagus, T. Honkela, S. Kaski and T. Kohonen, *Self-Organizing Maps of Document Collections: A New Approach to Interactive Exploration*, in Proceedings of the Second International Conference on Knowledge Discovery and Data Discovery 96, Portland (1996) pp. 238-243.

[21] J. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, Proc. 5th Berkeley Symp. Math. Stat. and Prob., 1 (1967) pp. 281-296.

[22] J. L. Marroquin and F. Girosi, *Some Extensions of the K-Means Algorithm for Image Segmentation and Pattern Classification*, A. I. Memo 1390, A. I. Laboratory, M.I.T. (1993).

[23] D. Merkl, E. Schwighofer, W. Winiwater, *Analysis of Legal Thesauri based on Self-Organizing Feature Maps*, *Proc. Fourth Internation Conference on Artificial Neural Networks*, University of Cambridge, UK (1995), pp. 227-235.

[24] C. E. Pykett, *Improving the Efficiency of Sammon's Nonlinear Mapping by using Clustering Archetypes* Electron. Lett. 14 (1978), pp. 799-800.

[25] R. Rao, J. O. Pedersen, M. A. Hearst, J. D. Mackinlay, S. K. Card, L. Masinter, P. K. Halvorsen, G. G. Robertson, *Rich Interaction in the Digital Library*, Communications of the ACM, 38 (1995), pp. 29-39.

[26] E. Rasmussen, *Clustering Algorithms*, in *Information Retrieval: Data Structures and Algorithms*, (eds. W. B. Frakes and R. Baeza-Yates), Prentice-Hall, (1992), pp.419-442.

[27] A. Ruhe and P. A. Wedin, *Algorithms for Separable Nonlinear Least Squares Problems*, SIAM Review, 22 (1980) pp. 318-336.

[28] G. Salton and C. Buckley, *Term weighting approaches in automatic text retrieval*, Information Processing and Management, 24 (1988), pp. 513-523.

[29] J. W. Sammon, *A Nonlinear Mapping for Data Structure Analysis*, IEEE Trans. Comput., 18 (1969) pp. 401-409.

[30] D. F. Shanno and K, H, Phua, *ALGORITHM 500: Unconstrained Minimization of Multivariate Function*, ACM Transactions on Mathematical Software, 1 (1975), pp. 87-94.

[31] P. Willett, *Recent Trends in Hierarchic Document Clustering: A Critical Review*, Information Processing and Management, 24 (1988), pp. 577-597.