



A dimensionality reduction algorithm and its application for interactive visualization

Jiyuan An^a, Jeffrey Xu Yu^b, Chotirat Ann Ratanamahatana^c,
Yi-Ping Phoebe Chen^{d,e,*}

^a*School of Information Technology, Faculty of Science & Technology, Deakin University, Melbourne, Australia*

^b*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, China*

^c*Department of Computer Science & Engineering, University of California, Riverside, CA 92521, USA*

^d*Faculty of Science & Technology, Deakin University, Melbourne, Australia*

^e*Australia Research Council Centre in Bioinformatics, Australia*

Received 27 November 2004; received in revised form 1 March 2006; accepted 11 March 2006

Abstract

Visualization is one of the most effective methods for analyzing how high-dimensional data are distributed. Dimensionality reduction techniques, such as PCA, can be used to map high dimensional data to a two- or three-dimensional space. In this paper, we propose an algorithm called HyperMap that can be effectively applied to visualization. Our algorithm can be seen as a generalization of FastMap. It preserves its linear computation complexity, and overcomes several main shortcomings, especially in visualization. Since there are more than two pivot objects in each axis of a target space, more distance information needs to be preserved in each dimension. Then in visualization, the number of pivot objects can go beyond the limitation of six (2-pivot objects \times 3-dimensions). Our HyperMap algorithm also gives more flexibility to the target space, such that the data distribution can be observed from various viewpoints. Its effectiveness is confirmed by empirical evaluations on both real and synthetic datasets.

© 2006 Elsevier Ltd. All rights reserved.

Keywords: Visualization; High-dimensional data; Dimensionality reduction

*Corresponding author.

E-mail addresses: jiyuan@deakin.edu.au (J. An), yu@se.cuhk.edu.hk (J.X. Yu), ratana@cs.ucr.edu (C.A. Ratanamahatana), phoebe@deakin.edu.au (Y.-P. Phoebe Chen).

1. Introduction

To help users understand and analyze datasets, visualization is a widely used technique [1]. From the visual data distribution, experts could find their clusters of interest with their professional knowledge. However, it is difficult to map high-dimensional data to two- or three-dimensional space while preserving the distance information. Many methods of dimensionality reduction have been proposed. Principle component analysis (PCA) [2] is a classical method, but with high dimensionality, it normally fails to select “good” axes in which the variance of coordinates is much larger than others. Many other visualization algorithms have been proposed, such as, Visdb [3,4], OPTICS [5], MDS [6], LLE [7] and Isomap [8]. However, the data distribution in the target space created by these algorithms only gives one form of representation. An expert then could only analyze the data distribution from this *only one* representation available in the target space; there is no room for users to change their viewpoints. And because there is a significant loss of information during the mapping onto a two- or three-dimensional space, it is difficult for all the clusters to be recognized in only one distribution in the target space.

In this paper, we focus on visualizing high-dimensional data in a two- or three-dimensional space, as well as in the simplest case of one-dimensional space. We introduce a parametric approach which allows users to interactively adjust their viewpoints in the original space and observe the changes in data distribution. Our parametric approach is developed on top of a novel mapping algorithm called *HyperMap*, which we also propose in this paper. The motivation of our work comes from an ideal situation where we could discover a way to prevent information loss during the dimensionality reduction process while data points are located in two- or three-dimensional space. Although these two conditions seem conflicting, HyperMap can alleviate them by adopting our concept of hyperaxis.

On a Euclidean space, a coordinate value of a data point on an axis can be viewed as the distance between its projection on the axis and the origin of the axis. Its sign is determined by observing its projection if it lies on the same side of the axis’ direction. We augment this approach in a way that if an axis is extended from a line to a hyperplane, any high-dimensional space can be represented by two- or three-dimensional space, consisting of our so-called *hyperaxes*. If the total number of hyperaxes is equal to the dimensionality of the original space, there will be no distance information loss. The number of dimensions of the hyperaxes can in fact be arbitrary, and are user-defined parameters in this paper. The coordinates can also be calculated from projections of data points in the hyperaxes. The predominant aim of this paper is to propose a method to compute coordinates in a new space. We will give the detailed explanation in later sections.

As a special case when all hyperaxes are one dimensional (a line), HyperMap is identical to FastMap algorithm [9]. In other words, HyperMap is a generalization of FastMap. In FastMap, two pivot objects are extracted one at a time to determine an axis. The axis of target space is then constructed dimension by dimension. However, FastMap suffers from two main drawbacks. Firstly, every pivot object is crucial for differentiating the data in the target space, such that a bad pivot object selection may lead to poor clustering quality. Secondly, the target space has only one form of representation, in which users cannot interactively observe the distribution of data in the visual space. In contrast to FastMap, an axis in HyperMap consists of k (≥ 2) pivot objects, which can be represented by $(k-1)$ -dimensional hyperaxes. The parametric approach developed on top of HyperMap, allows

users to interactively observe the data distribution in two- or three-dimensional space. Similar to FastMap, HyperMap also selects pivot objects in linear time. Our research achievements, detailed in this paper, are given below:

- We define a *hyperaxis* and a *coordinate* of hyperaxis, which helps to overcome the limitation of Euclidean space that an axis is only represented as a line. In our approach, an axis can be represented as a line, a plane, or a hyperplane.
- By introducing *relative coordinates*, we derive a formula that converts coordinates from an original space to a target space.
- We develop an interactive technique that allows users to change their viewpoints by tuning the weight associated with each hyperaxis. By varying the weights, users can interactively see the data distribution from different viewpoints.

The rest of the paper is organized as follows. Section 2 addresses definitions and preliminaries. Section 3 outlines our parametric approach. The analysis of HyperMap is demonstrated in Section 4. We conclude the paper in Section 5.

2. Preliminaries

Most of the datasets can be viewed as cluster data, such as web documents and genomic data. In order to effectively understand a specific web document, it is desirable to know what cluster it belongs to. The number of keywords in a document is usually large and the documents, therefore, correspond to points in a very high-dimensional space. Visualization is the representation of the documents as data points in two- or three-dimensional space. For a new-coming web document, we also map it as a data point in the two- or three-space so that we can subsequently determine which cluster it belongs to. The crucial problem is how to reduce the dimensionality without much loss of distance between two arbitrary data points. Dimension reduction has been well-studied until recently. The HyperMap is the first algorithm to give flexible target space. Here, we give an example to explain our visualization method; HyperMap. Principle components analysis (PCA) is a widely used dimension reduction method. However, it is limited by a fixed target space. In our investigation, we will demonstrate its ineffectiveness in visualization using a synthetic dataset.

Fig. 1 shows the two-dimensional target spaces for 10-dimensional datasets that comprise five clusters: each cluster has 100 records. The two axes of the target space are represented by the first and second components, respectively. Only the fifth cluster is separated from the other four clusters. When the first and third components are selected as the axes of target space, as shown in Fig. 2, the fourth cluster can be found. By selecting different components, the clusters can be separated from each other. However, it will make the process of visualization very laborious for the following reasons:

- We have no knowledge about the quality of components and as a result, it is difficult to determine whether they are “good” and “bad”.
- As combinations of components for building a target space are numerous, enumeration of all possible combinations is impracticable. For example, 20-dimensional datasets have 20 components; the combination of two- and three-dimensional target spaces are 190 and 1140, respectively.

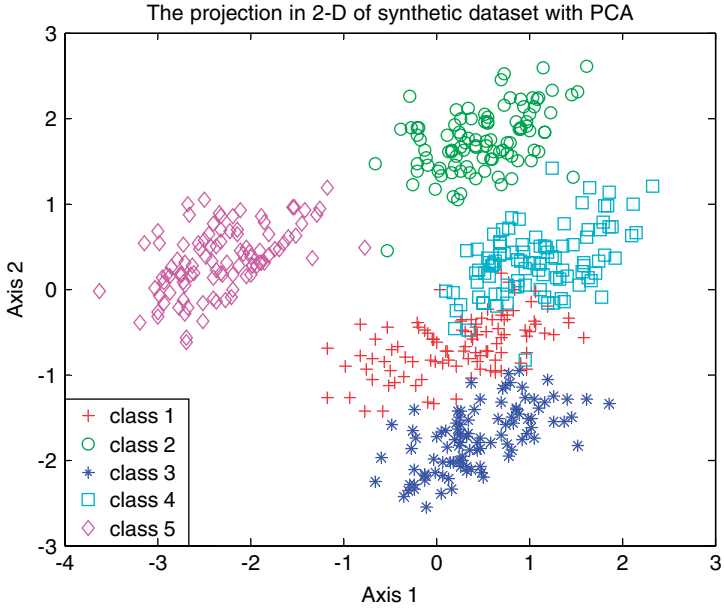


Fig. 1. Visualization of 2-D target space. Axes 1 and 2 are the first and second components of the dataset.

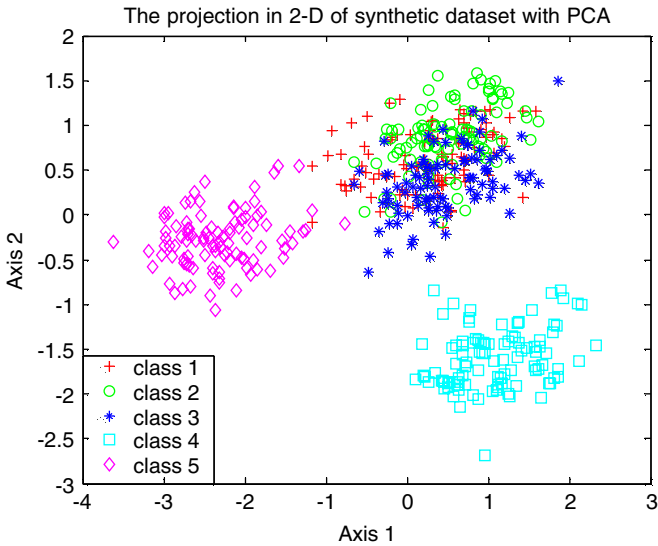


Fig. 2. Visualisation of 2-D target space. Axes 1 and 2 are the first and third components of the dataset.

HyperMap provides only ONE target space, but users can change their viewpoints to get different data distribution and subsequently find the clusters. This differentiates HyperMap from other proposed visualization methods.

Table 1
Notations used in this paper

Symbol	Meaning
O	Dataset of objects $\{o_i\}$
N	Number of objects in O
K	Number of pivot objects of a hyperaxis
p_i	One of pivot objects that constitutes hyperaxis
w_i	One of the weights of a hyperaxis
l_i	Distance between object p_i 's projection and p
$p_1p_2 \dots p_n$	Hyperaxis determined by p_1, p_2, \dots, p_n
W	A sequence of weights (w_1, w_2, \dots, w_n) for a hyperaxis
$\overline{p_1p_2 \dots p_n}$	Complementary space perpendicular to $p_1p_2 \dots p_n$
$o p_1p_2 \dots p_n$	Projection of point o onto a hyperaxis $p_1p_2 \dots p_n$
$dist(p, q)$	Distance between p and q
$\overline{dist}(p, q)$	Projected distance in complementary space
$L(o, p_1p_2 \dots p_n)$	A sequence of distance (l_1, l_2, \dots, l_n) between projection of o and pivot objects
$D(o, p_1p_2 \dots p_n)$	The relative coordinate for o on hyperaxis $p_1p_2 \dots p_n$
$o \cdot p_1p_2 \dots p_n$	o 's hypercoordinate on hyperaxis p_1, p_2, \dots, p_n

The term ‘‘hyperplane’’ is usually used for classification [10]. It means an $(n-1)$ -dimensional subspace of an n -dimensional space: $\{x \in \mathcal{R}^n, \langle x, a \rangle = b\}$, where $\langle \cdot, \cdot \rangle$ represents Euclidean inner product and a, b are constants. In this paper, ‘‘hyperplane’’ is extended to k -dimensional subspace in an n -dimensional space, where k is an integer from 1 to $n-1$. Table 1 summarizes the notations and symbols used in this paper.

The algorithm of HyperMap is used to map objects onto a lower-dimensional space. Since its axes are hyperplanes, we call it a virtual space. To avoid confusion with the common meaning of axis, we have introduced a novel concept of hyperaxis to indicate a hyperplane. Hyperaxis is determined by n objects (we call these pivot objects p_1, p_2, \dots, p_n). We use notation $p_1p_2 \dots p_n$ to indicate the hyperaxis. In addition, given an object o in original space, we use $o|p_1p_2 \dots p_n$ to denote the projection of o onto a hyperaxis $p_1p_2 \dots p_n$.

3. Our approach

We use an example shown in Fig. 3 to illustrate our motivation towards interactive visualization. If we allow only one-dimensional target space, it is impossible to separate these three clusters using PCA or FastMap. However, we can apply our *HyperMap* algorithm to recognize each cluster individually. We assume that pivot objects are at the center of each cluster (we will explain how to select pivot objects in Section 3.3). For each object, we calculate three distances between the current object and the three pivot objects, denoted as l_1, l_2 , and l_3 . The coordinates of the hyperaxis are calculated by the following formula.

$$x = w_1l_1 + w_2l_2 + w_3l_3. \quad (1)$$

A user can interactively tune the weights w_1, w_2 , and w_3 to visualize the three clusters individually. For instance, by setting $w_1 = 0.0, w_2 = w_3 = 0.5$, cluster C_1 is recognized as shown in Fig. 3(B), because the coordinate values of the data points in C_1 are larger than

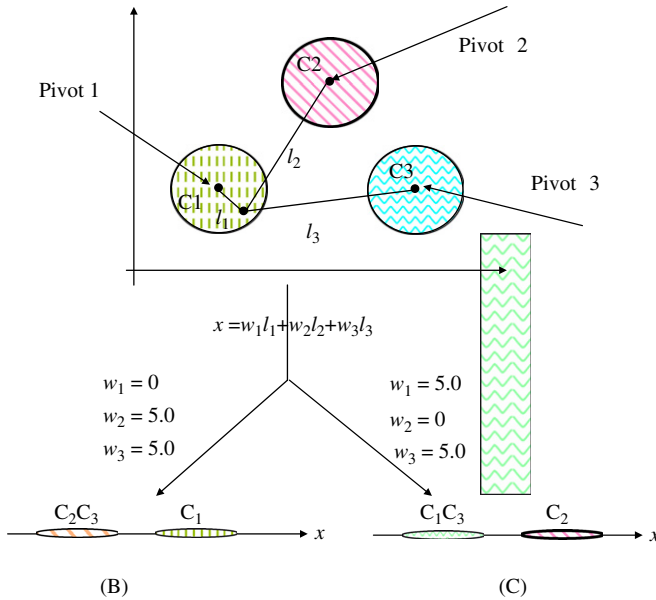


Fig. 3. There are three clusters in 2-D space. By tuning the weights of the hyperaxis, three clusters can be individually recognized.

those of the data points in other clusters; such as C_2 and C_3 . Similarly, by setting $w_1 = 0.5$, $w_2 = 0.0$, $w_3 = 0.5$, cluster C_2 is separated from the other two clusters as shown in Fig. 3(C).

Fig. 1, as above, is an example of a one-dimensional target space. We can also map onto two-, three-, or any k -dimensional target space. In the next section, we will explain how to construct a two-dimensional target space, which can easily be extended to other dimensional target spaces.

Firstly, two hyperplanes are determined by two respective sets of pivot objects. The size of the sets is the parameters to be given by the user. Then, data points are projected onto the two hyperplanes. Finally, on each hyperplane, coordinates on the target space can be calculated in the similar fashion as the one-dimensional target space shown in Fig. 3. As a result, in the three-dimensional target space, the coordinates of three hyperaxes are calculated from the following formula:

$$\begin{aligned}
 x &= w_1 l_1 + w_2 l_2 + \dots, \\
 y &= w'_1 l'_1 + w'_2 l'_2 + \dots, \\
 z &= w''_1 l''_1 + w''_2 l''_2 + \dots,
 \end{aligned}$$

where (x, y, z) is an object's coordinate in the target space. We can also generate various viewpoints by altering the weights $(w_1, w_2, \dots, w'_1, w'_2, \dots, w''_1, w''_2)$, which is an important aspect for parametric visualization in our work.

3.1. An example of constructing a 2-D target space

To best illustrate our HyperMap algorithm, we assume a three-dimensional original space. We use two-dimensional space for visualization. Given a three-dimensional original space, as shown in Fig. 4, we select three pivot objects (p_1, p_2, p_3) to determine hyperaxis 1 (a plane), and two pivot objects (q_1, q_2) to determine hyperaxis 2 (a line). To construct a virtual space, each object o is projected onto hyperaxes 1 and 2 as $o|p_1p_2p_3$ and $o|q_1q_2$, respectively. Its location can be represented by

$$(L(o, p_1p_2p_3), L(o, q_1q_2)) = ((l_1, l_2, l_3), (l'_1, l'_2)).$$

Fig. 4 illustrates an example of various object positions in different target spaces, according to user-defined weights W .

Since our target space is a Euclidean space, real numbers are used to represent an object’s location on each hyperaxis. To compute a coordinate, we apply a linear combination of distances between pivot objects and the projections of an object on the hyperaxis are then determined by these pivot objects. Similar to our definition of hyperaxis, we would like to avoid the common meaning of *coordinate* by introducing a novel concept called a *hypercoordinate*. The *hypercoordinate* indicates a coordinate on a hyperaxis; its definition is given below:

Definition 1. (hypercoordinate) For a hyperaxis $p_1p_2\dots p_k$, we assume that its weight $W = (w_1, w_2, \dots, w_k)$. Given an object o , if the distances between its projection and pivot

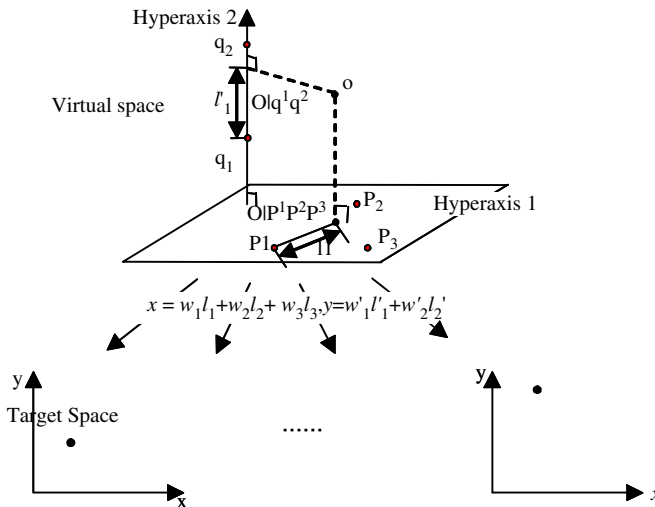


Fig. 4. Virtual space and target space. Converting from virtual space of a 2-D hyperaxis to a target space; axes X and Y in the target space correspond to hyperaxes 1 and 2, respectively. By varying the weights W , different target spaces are generated. The target space shows different data distribution according to different weights.

objects is denoted by $L = (l_1, l_2, \dots, l_k)$, the hypercoordinate of o on the hyperaxis is defined as

$$o \cdot p_1 p_2 \dots p_k = W \cdot L^T = \sum_{i=1}^k w_i \cdot l_i = \sum_{i=1}^k w_i \cdot \text{dist}(o|p_1 p_2 \dots p_k, p_i). \tag{2}$$

Here, $o|p_1 p_2 \dots p_k$ is the projection of object o on the $(k-1)$ -dimensional hyperaxis. l_i indicates the distance between $o|p_1 p_2 \dots p_k$ and the pivot object p_i ($1 \leq i \leq k$). w_i is a weight element that satisfies the following condition:

$$|w_1| + |w_2| + \dots + |w_k| = 1. \tag{3}$$

The constraint in Eq. (3) ensures that the distance between two hypercoordinates $o_1 \cdot p_1 p_2 \dots p_k$ and $o_2 \cdot p_1 p_2 \dots p_k$ cannot be larger than the distance between $o_1|p_1 p_2 \dots p_k$ and $o_2|p_1 p_2 \dots p_k$, which is the $\text{dist}(o'_1, o'_2)$ shown in Fig. 5. This is simply because the hypercoordinates are “extracted” from the corresponding hyperplane (hyperaxis). We give our proof below using the triangle inequation. For simplicity, we use the notations of o'_1 and o'_2 for the projections of $o_1|p_1 p_2 \dots p_k$ and $o_2|p_1 p_2 \dots p_k$, respectively.

$$\begin{aligned} |o_1 \cdot p_1 p_2 \dots p_k - o_2 \cdot p_1 p_2 \dots p_k| &= |w_1(\text{dist}(o'_1, p_1) - \text{dist}(o'_2, p_1)) \\ &\quad + w_2(\text{dist}(o'_1, p_2) - \text{dist}(o'_2, p_2)) \\ &\quad + w_k(\text{dist}(o'_1, p_k) - \text{dist}(o'_2, p_k))| \\ &\leq (|w_1| + |w_2| + \dots + |w_k|) \max_{1 \leq m \leq k} |(\text{dist}(o'_1, p_m) - \text{dist}(o'_2, p_m))| \\ &\leq \text{dist}(o'_1, o'_2). \end{aligned}$$

Unlike α -mapping [11] that employs 2D *star coordinates* [12] to establish the visualization, HyperMap guarantees that the distances between two data points in the target space will not be bigger than that in the original space.

3.2. HyperMap algorithm

Each hyperaxis of virtual space is determined by pivot objects, whose number is a user-defined parameter. Without loss of generality, we assume that the number of pivot objects

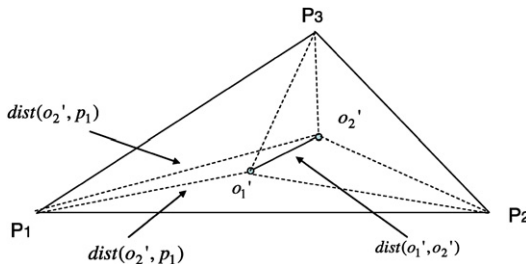


Fig. 5. The distance between o_i and o_j 's hypercoordinates does not exceed the distance of their projection on the hyperaxis, that is, $|\text{dist}(o'_i, p_k) - \text{dist}(o'_j, p_k)| \leq \text{dist}(o'_i, o'_j)$ ($1 \leq k \leq 3$). Notice that all the points in the figure are on a plane.

is k . The algorithm of HyperMap can be divided into five steps. The five steps are as follows.

1. *Pivot selection*: Determine a $(k-1)$ -dimensional hyperaxis by selecting k pivot objects p_1, p_2, \dots, p_k . For example, we can select three pivot objects, p_1, p_2, p_3 , to generate a two-dimensional plane as a hyperaxis.
2. *Computing hypercoordinate*: From our Definition 1 of hypercoordinate, L should first be calculated; L is defined as $l_i = \text{dist}(o|p_1, p_2, \dots, p_k p_i)$. Recall that $o|p_1 p_2 \dots p_k$ is the projection of object o on the hyperaxis. When the hyperaxis is more than two dimensional, finding the projection of a point on a hyperaxis is difficult. Therefore, it is more complicated to calculate the distances l_1, l_2, \dots, l_k between a pivot object and the projection of point o . In Section 3.4, we propose an algorithm to resolve this problem by introducing the novel concept of *relative coordinate*.
3. *Preparation for computing hypercoordinates of subsequent hyperaxis*: To calculate the hypercoordinates of the subsequent hyperaxis, it is necessary to calculate projected distances between all pairs of objects in the complementary space, $\overline{p_1 p_2 \dots p_k}$ which is perpendicular to the hyperaxis $p_1 p_2 \dots p_k$. The formula and algorithm details are given in Section 3.4.
4. *Looping*: The ability to compute the projected distance in complementary space allows us to project on a second hyperaxis, that lies on the complementary space $\overline{p_1 p_2 \dots p_k}$. Within this space, we can recursively apply steps 1–3 until all hyperaxes of virtual space are obtained. The maximum number of hyperaxes is 3, because it is not easily visualized using 2-D displays.
5. *Hypercoordinate calculation*: Compute the hypercoordinates of the hyperaxis for all data points using Eq. (2).

In the case where $k = 2$, there are two pivot objects (p_1 and p_2). If we assume that the weights for two pivot objects are w_1 and w_2 , for every data point o , its hypercoordinate is $w_1 * \text{dist}(o|p_1 p_2, p_1) + w_2 * \text{dist}(o|p_1 p_2, p_2)$. If $w_2 = 0$, HyperMap is specified to FastMap [9]. In other words, FastMap algorithm is a special case of HyperMap.

3.3. Pivot selection

In order to select the k pivot objects for large high-dimensional datasets in linear time, we exploit the k -center algorithm given in [13], which selects k well separated objects in a high dimensional space in linear time.

As shown in Table 2 above, the pivot selection algorithm includes the following two steps:

1. Randomly select an object $o_i \in o$. The first pivot object p is selected as the object that has the maximum distance from o_i , and is added into pivot object set S which is initially empty.
2. Repeatedly find a pivot object p that satisfies the following property:

$$\min_{o \in S, o' \in O} (\text{dist}(o, o')) \leq \min_{o \in S} (\text{dist}(o, p)) \quad (4)$$

until all k pivot objects are selected.

Table 2
Pivot selection algorithm

Algorithm: Select pivot (o , $dist$, $()$, k)

Output: A set of k objects 2 :

Begin

Randomly choose an object o from o .

Choose p such that for any o' in o ,

$dist(o, o') \leq dist(o, p)$

Add p to S

While $|S| < k + 1$ **do**

Choose next pivot object p from

Eq. (4)

Add p to S

end while

return S

End

Note that similar to the selection algorithm of pivot objects in FastMap, the computation complexity of this algorithm is also linear.

3.4. Computation of hypercoordinate

To find the location of projection o' for object o on the hyperaxis $p_1p_2\dots p_k$ ($(k-1)$ -dimensional plane), $k-1$ real numbers are usually needed, because the points $p_1p_2\dots p_k$ are always linearly independent. For example, to express a point in a three-dimensional space, three real numbers are necessary, which are simply its coordinates, (x,y,z) . To be able to express a point on a hyperplane in HyperMap algorithm, we must first introduce a novel concept called *relative coordinate*, defined below:

Definition 2. (relative coordinate) Given a $(k-1)$ -dimensional hyperaxis $p_1p_2\dots p_k$ ($k > 1$), the i th relative coordinate of an object o , with respect to a hyperaxis $p_1p_2\dots p_k$, has two parts, i.e. its absolute value and its sign: the absolute value of object o 's relative coordinate is the value of distance between $o|p_1p_2\dots p_i$ and an $(i-1)$ -dimensional hyperplane $p_1p_2\dots p_{i+1}$ ($2 \leq i \leq k-1$), which is determined by i pivot objects p_1, p_2, \dots, p_i . The sign of the relative coordinate is: if the object o is on the same side as the previous pivot object, p_{i+1} , with respect to the hyperplane $p_1p_2\dots p_i$, its relative coordinate is positive, otherwise it is negative.

$k-1$ relative coordinates are denoted as $D(o, p_1p_2)$, $D(o, p_1p_2p_3)$, and $D(o, p_1p_2\dots p_k)$. In the following sections, we will explain how to calculate the distance l_1, l_2, \dots, l_k between the projection and pivot objects, and projected distance of two objects in complementary space using relative coordinates. Fig. 6 (top) provides the definition for relative coordinate. Data point o has a positive value of relative coordinate, while data point o' has a negative value. It is because $O|p_1p_2\dots p_k$ is on the same side with p_k , in terms of hyperplane $p_1p_2\dots p_k$, but $O'|p_1p_2\dots p_k$ is on the opposite side. In general, each object o has $k-1$ relative coordinates, $D(o, p_1p_2)$, $D(o, p_1p_2p_3), \dots, D(o, p_1p_2\dots p_k)$. In the specific instance; $D(o, p_1) = 0$, It is because that the distance between o 's projection in p_1 and p_1 is zero.

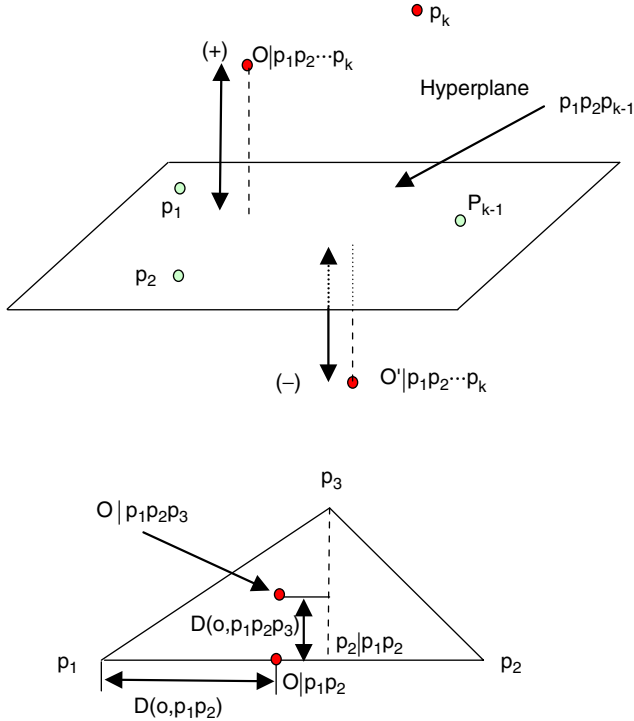


Fig. 6. Definition of relative coordinate of hyperaxis. (Top) illustrates $D(o, p_1p_2 \dots p_k)$ in a $(k-1)$ -dimensional hyperaxis $p_1p_2 \dots p_k$. (Bottom) a two-dimensional hyperaxis $p_1p_2p_3$. An object o 's relative coordinates $D(o, p_1p_2)$ and $D(o, p_1p_2p_3)$ are illustrated.

We find that the relative coordinates can be calculated recursively from $D(o, p_1p_2)$ to $D(o, p_1p_2 \dots p_k)$.

In the following subsections, we describe the detailed computation processes required to obtain the objects' hypercoordinates.

3.4.1. Calculating relative coordinate

Relative coordinate can be calculated by firstly computing $D(o, p_1p_2)$ for one-dimensional hyperaxis, then computing $D(o, p_1p_2 \dots p_{k+1})$ from $D(o, p_1p_2 \dots p_k)$.

1. *Computing $D(o, p_1p_2)$* : For simplicity, we demonstrate how to calculate relative coordinates in the simplest case of one-dimensional hyperaxis. We use o' (or $o|p_1p_2$) to indicate the projection of point o in one-dimensional hyperaxis and as shown in Fig. 7. An object o is projected onto the one-dimensional hyperaxis (a line) p_1p_2 . The o 's relative coordinate $D(o, p_1p_2)$ in the axis can be computed by

$$D(o, p_1p_2) = \text{dist}(o', p_1) = \frac{(\text{dist}(o', p_1))^2 - (\text{dist}(o', p_2))^2 + (\text{dist}(p_1, p_2))^2}{2 \text{dist}(p_1, p_2)}. \tag{5}$$

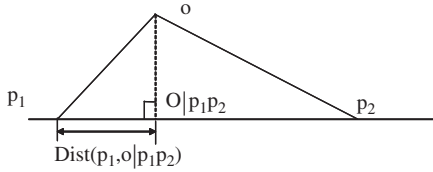


Fig. 7. For one-dimensional hyperaxis (line), there is only one relative coordinate $D(o, p_1 p_2)$, except for $D(o, p_1)$ (its value is zero).

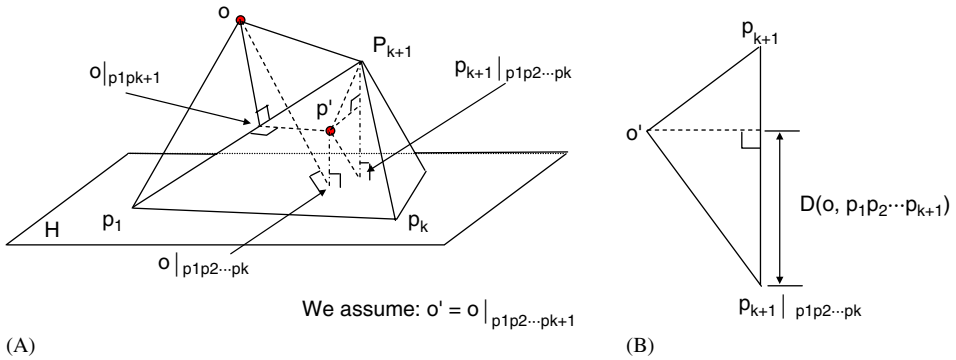


Fig. 8. Relative coordinate computed in a k -hyperaxis.

2. *Computing $D(o, p_1 p_2 \dots p_{k+1})$ from $D(o, p_1 p_2 \dots p_k)$:* We compute the relative coordinate $D(o, p_1 p_2 \dots p_k)$ using an inductive method. Fig. 8 shows a hyperaxis formed by $k+1$ pivot objects. The formulation is given in Table 3. The simplest case is 2-D hyperaxis’s relative coordinate $D(o, p_1 p_2)$. To completely understand Table 3, we list the computing method of 2-D hyperaxis’s relative coordinate in the appendix A.

3.4.2. *Calculating height from a data point to a hyperplane*

During the mapping of each object to a point on the target space, hypercoordinates of objects need to be calculated. As a part of this process, the height (distance) from a data point to a hyperplane must also be known; this is proposed in Lemma 1.

Lemma 1. Given a data point o , it has a projection o' in hyperaxis $p_1 p_2 \dots p_k$. The vertical distance h from o to the hyperaxis can be calculated from the following equation:

$$h = \text{dist}(o, o') = \sqrt{(\text{dist}(o, p_1))^2 - \sum_{i=2}^k D^2(o, p_1 p_2 \dots p_i)} \tag{7}$$

We can prove Lemma 1 recursively by focusing firstly on the line $p_1 p_2$, the distance $D(o, o|_{p_1 p_2})$ can be calculated simply as shown in Fig. 9.

Then, having $D(o, o|_{p_1 p_2})$ and $D(o|_{p_1 p_2}, o|_{p_1 p_2 p_3})$, we can compute $D(o, o|_{p_1 p_2 p_3})$. Similarly, we provide proof for $h = D(o, o|_{p_1 p_2 \dots p_k})$ below.

Proof. As shown in Fig. 9, and from the definition of relative coordinate,

$$\begin{aligned} (dist(o, p_1))^2 &= (dist(p_1, o|_{p_1p_2}))^2 + (dist(o, o|_{p_1p_2}))^2 \\ &= D^2(o, p_1p_2) + (dist(o, o|_{p_1p_2}))^2, \\ \therefore o|_{p_1p_2} \ o|_{p_1p_2p_3} &\text{ is on the hyperplane } p_1p_2p_3, \\ \therefore o \ o|_{p_1p_2} &\perp o|_{p_1p_2} \ o|_{p_1p_2p_3}, \\ (dist(o, o|_{p_1p_2}))^2 &= (dist(o|_{p_1p_2}, o|_{p_1p_2p_3}))^2 + (dist(o, o|_{p_1p_2p_3}))^2 \\ &= D^2(o, p_1p_2p_3) + (dist(o, o|_{p_1p_2p_3}))^2. \end{aligned}$$

In general, we have,

$$(dist(o, o|_{p_1p_2 \dots p_i}))^2 = D^2(o, p_1p_2 \dots p_{i+1}) + (dist(o, o|_{p_1p_2 \dots p_{i+1}}))^2.$$

As a result,

$$\begin{aligned} dist(o, p_1)^2 &= \sum_{i=2}^k D^2(o, p_1p_2 \dots p_i) + (dist(o, o|_{p_1p_2 \dots p_k}))^2 \\ &= \sum_{i=2}^k D^2(o, p_1p_2 \dots p_i) + (dist(o', o))^2. \end{aligned}$$

3.4.3. Calculating projected distance in the complementary space

To map data distribution onto a two- or three-dimensional target space, two or three corresponding hyperaxes need to be constructed. By using relative coordinates, each object's projection in the first hyperaxis can be computed by the method explained in the previous section. However, to calculate an object's projection on the second or other hyperaxes, the distance of all object pairs in the complementary space must first be computed. Fig. 10 illustrates a method by which to compute $\overline{dist}(o_1, o_2)$, which is the projected distance of two objects o_1 and o_2 in complementary space of hyperaxis p_1p_2 .

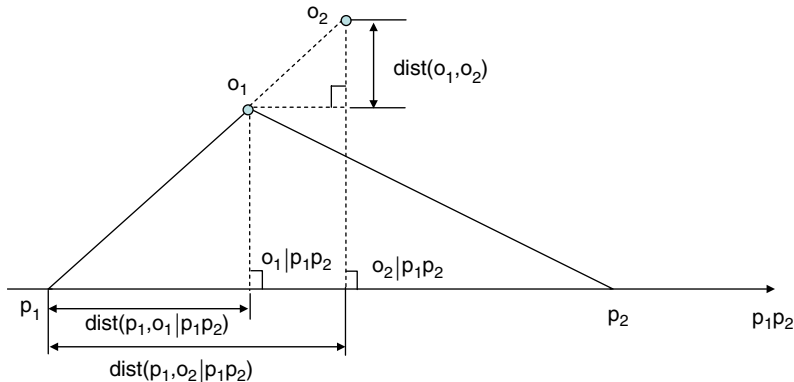


Fig. 10. In the case of one-dimensional hyperaxis (line), projected distance between data points, o_1 and o_2 in the complementary space of hyperaxis p_1p_2 is calculated.

In the case of one-dimensional hyperaxis, we have:

$$\overline{dist}(o_1, o_2) = \sqrt{(dist(o_1, o_2))^2 - (D(o_1, p_1 p_2) - D(o_2, p_1 p_2))^2}. \quad (8)$$

The relative coordinate $D(o, p_1 p_2 \dots p_k)$ and projected distance in complementary space $\overline{dist}(\cdot, \cdot)$ can be computed by using an inductive method.

Given two objects o_1 and o_2 , the projected distance between o_1 and o_2 in the complementary space can be computed by the following equation:

$$\overline{dist}(o_1, o_2) = \sqrt{(dist(o_1, o_2))^2 - \sum_{i=2}^k (D(o_1, p_1 p_2 \dots p_i) - D(o_2, p_1 p_2 \dots p_i))^2}. \quad (9)$$

3.4.4. Hypercoordinate calculation

Finally, when all the necessary calculations that have been discussed in Sections 3.4.1–3.4.3 are completed, we are then able to compute the hypercoordinates of all the data points in the target space. Their location L on every hyperaxis is now defined (Eq. (2)). Applying the Pythagorean theorem, we have

$$l_i = \sqrt{dist(o, p_i)^2 - h^2}.$$

And from Lemma 1, we have

$$l_i = \sqrt{(dist(o, p_i)^2 - (dist(o, p_1))^2 + \sum_{j=2}^k (D^2(o, p_1 p_2 \dots p_j))}. \quad (10)$$

3.5. HyperMap algorithm

Our HyperMap algorithm is outlined in Table 4. The complexity of the HyperMap algorithm is $O(N\bar{k})$ for constructing one hyperaxis, where \bar{k} is an average value for the number of pivots.

From lines 5 to 6, the relative coordinates are calculated. Relative distance calculation is a crucial step in our algorithm; it allows us to compute hypercoordinates and the distance of all pairs of objects in complementary space. By replacing $dist(\cdot, \cdot)$ with $\overline{dist}(\cdot, \cdot)$, the relative coordinates $D(\cdot, \cdot)$ in the complementary space can now be computed with respect to the data points and pivot objects.

4. Experimental evaluation

To evaluate a method of dimensionality reduction, stress function is typically used. It evaluates the loss of information during the transformation from the original dimensionality to a target space. The stress measure is defined in Eq. (11) below, where $dist(i, j)$ is the distance between the two objects o_i and o_j in target space. $dist(i, j)$ denotes

Table 4
HyperMap Algorithm

Algorithm HyperMap($n, dist(), O$)
Global variables: $D(o, p_1 p_2 \dots)$
 {saving relative coordinate to each pivot objects}
 $npivot[1], npivot[2], \dots, npivot[n]$
 {Number of pivot objects at each level}
Input:
 n {Number of pivot objects}
 $\overline{dist}(\cdot, \cdot)$ {Distance function}
 O {Dataset}
Output:
 $o.coor[leve \#][pivot \#]$
 {distance from data to each pivot object, w.r.t. level}
Begin:
 1. **if**($n \leq 0$) then **Return**
 2. Select-pivot($O, dist(), npivot[n]$){cf. Table 2}
 3. **if**($\forall_i \forall_j dist(p_i, p_j) = 0$) then **Return** //dimensions run out
 4. **For each** data o in O **do**
 5. **For each** pivot p_j ($j = 2:npivot[n]$) **do**
 {Relative coordinates are calculated (cf. Table 3)}
 calculate $D(o, p_1 p_2 p_j)$
 6. **End for**
 {compute distance from o to hyperaxis (Lemma 1)}
 7. $h^2 = (dist(o, p_1))^2$
 8. **For** $t = 1:npivot[n]$ **do**
 9. $h = h - D(o_i, p_1 p_2, \dots, p_t)^2$
 10. **End for**
 11. $h = \sqrt{h}$
 12. **For each** pivots p_j ($j = 1:npivot[n]$) **do**
 13. $o.coor[n][j] = \sqrt{(dist(o, p_j))^2 - h^2}$
 14. **End for**
 15. **End for**
 {compute distances for each pair of data points using Eq. (9)}
 16. **For each** data o_i in O **do**
 17. **For each** data o_j in O **do**
 18. $dist'(o_i, o_j) = (dist(o_i, o_j))^2$
 19. **For** $t = 1:npivot[n]$ **do**
 20. $dist'(o_i, o_j) = dist(o_i, o_j) - (D(o_i, p_1 p_2, \dots, p_t) - D(o_i, p_1 p_2, \dots, p_t))^2$
 21. **End for**
 22. $dist'(o_i, o_j) = \sqrt{dist'(o_i, o_j)}$
 23. **End for**
 24. **End for**
 25. HyperMap($n-1, dist(), O$)

the distances between the two objects in the original space. The smaller the stress value, the less the information has been lost.

$$stress = \sqrt{\frac{\sum_{i,j} (dist'(i,j) - dist(i,j))^2}{\sum_{i,j} (dist(i,j))^2}} \tag{11}$$

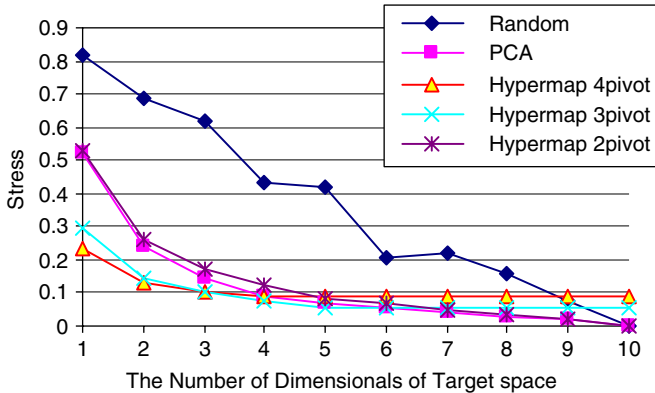


Fig. 11. The stress of different dimensionality reduction methods.

The main difference among various dimensionality reduction methods is the axes selection technique of a target space. The better chosen the axis, the more the distance information is preserved in the target space. To evaluate our approach, we compare the stress value of our method with those of other existing methods.

One simple method is to select axes randomly, although the result may not be satisfactory. Another classical method such as PCA, first calculates the eigenvector of the covariance matrix. Then the k th “good” axes are selected to be part of the k -dimensional target space. Note that for n -dimensional data, there will be no loss of information if $k = n$.

If HyperMap selects two pivot objects for every hyperaxis, HyperMap becomes identical to FastMap. Fig. 11 shows the stress values of target spaces whose axes were selected randomly by various approaches, PCA and HyperMap algorithm. With an increase of dimensionality in the target space, the distance-information loss further reduces. HyperMap is tested with several numbers of pivot objects. More specifically, two, three, and four pivot objects were tested on the 10-dimensional synthetic data (see Section 4.1.1 for the dataset details).

We can see from the figure that when 10 axes are selected, all “Random”, “PCA” and “HyperMap 2Pivot”(= FastMap) have 0 stress values, while “HyperMap 3Pivot” and “HyperMap 4Pivot” have a stable stress value when the number of axes is more than 2; their stress values never converge to 0. This is not surprising because their hyperaxes are not lines; the distance information has already been lost during the coordinate calculations. This supports our initial motivation; our main objective is not to try to preserve all the distance information in the complementary space, instead we only need to keep as much distance information as possible in the first few dimensions.

From our preliminary evaluation, we can suggest that our HyperMap dimensionality reduction method does preserve distance information well in the first several dimensions. Other than integrating HyperMap into an interactive visualization system, as will be illustrated in the next section, HyperMap also provides us with a promising possibility to index the reduced dimensionality data from a high-dimensional dataset, without much loss of distance information.

4.1. Visualization for synthetic and real data

Using visualization to find clusters in a large dataset has proven to be a very practical and useful method in analyzing the data. However, there are many diverse definitions for a cluster itself; in some cases, it is defined as a collection of objects densely grouped together in a ball shape, but in most cases, this “well defined” definition is unfeasible. According to [5,14–16], we can regard data points that centralize around a hyperplane to be a cluster. Or we can perceive a torus-shaped data distribution to also be a cluster. In this paper, we show that data distribution can be displayed with different viewpoints, such that the users can give the final evaluation of clusters by observing data contributions from different angles.

In this paper, we utilize the HyperMap algorithm in order to develop a data mining tool called *interactive visualization*. HyperMap enables users to enhance their visualization of high dimensional data distributions by adjusting the weights and the number of pivot objects on each hyperaxis. However, users need to provide the algorithm with these two parameters, i.e. the number of pivot objects on the hyperaxes, and the weights for hypercoordinate calculation in the target space. Our approach towards parameter selection is as follows:

- We can find the appropriate number of pivot objects (= the number of weights) from their stress values. For example in Fig. 11, in a three-dimensional target space, the differences among “HyperMap 2”, “HyperMap 3”, and “HyperMap 4” are very subtle. Hence, the simplest representation of “HyperMap 2” should be a satisfactory choice, i.e. two pivot objects for three hyperaxes.
- To get a better data distribution in the target space, larger hypercoordinates are desirable. In other words, $w_1 \times \sum l_1 + w_2 \times \sum l_2 + \dots + w_d \times \sum l_d$ should be large. From the weight constraint condition in Eq. (3), we can maximize the expression by simply setting one of the weights with the largest summation term to be 1 and the rest of the weights to be 0. In the case where the number of pivot objects is 3, only 9 (3×3) test evaluations are necessary before we discover the largest data distribution. Our visualization tool is very easy to use, and also enables the users to easily observe data distribution from different viewpoints by tuning the weights.

We tested our method on both synthetic and real data, which will be discussed in the next two subsections.

4.1.1. Synthetic data

To generate a dataset, we used a method similar to the one discussed in [17]. In this method, the anchor points of clusters are first determined. Then, the number of points associated with each anchor point is determined, and finally cluster points are generated.

More specifically, the anchor points of clusters are obtained by generating k uniformly distributed points in d -dimensional space of $[0.0-1.0]$. All clusters have the same number of points. The positions of the data in each cluster follow the normal distribution, with the anchor point as its mean and variance μ . In our experiments, the size of the dataset is 1000 data points divided into five clusters, i.e. $k = 5$. We set the number of clusters to $k = 5$, and the variance of all clusters to $\mu = 0.2$. To illustrate the characteristic of the data, we selected two pivot objects to be represented in three hyperaxes. The weights are set to be

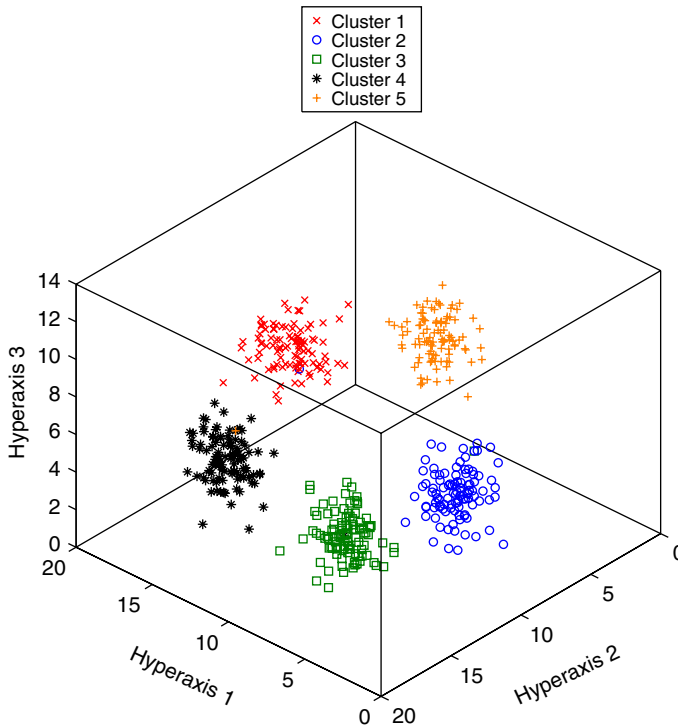


Fig. 12. Synthetic dataset shown in 3-D target space.

$W = \{1,0;1,0;1,0\}$. Fig. 12 shows the data distribution of the synthetic dataset in a three-dimensional target space.

The stress of this synthetic data is shown in Fig. 11. With the exception of the “Random” method, the distance information of all other methods appears to be well-preserved in the three-dimensional target space.

4.1.2. Real dataset

In [9], Faloutsos and Lin evaluated their FastMap algorithm on WINE dataset {<http://kdd.ics.uci.edu/>}, and discovered that “the 3-D distribution-plot gives the whole picture and separates the clusters almost completely”. We also evaluated our HyperMap algorithm on this real dataset. The results of the stress test are shown in Fig. 13.

Evidently, there is not much difference between FastMap algorithm in 3-D and our HyperMap-3Pivot in 2-D. As a result, we are able to achieve good data distribution in a two-dimensional target space, with the number of pivot objects or weight of 3. As expected, when we tuned the weights according to Table 5, all three clusters appear to be well-separated in two-dimensional space, as shown in Fig. 14. Note that the values in Table 5 are only an example of a set of weights that give good visualization. These numbers can also be obtained from our visualization tool. The demonstration of the parametric visualization is available at (<http://www.dblab.is.tsukuba.ac.jp/~an/HyperMap/HyperMapDemo.html>).

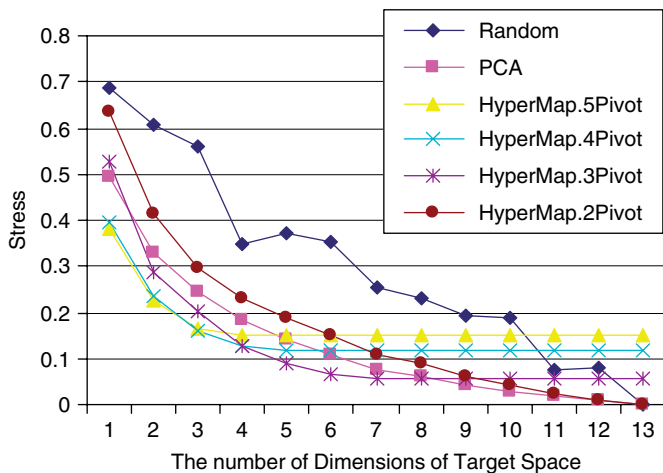


Fig. 13. The stress for wine dataset.

Table 5
Weights used in our experiment

Hyperaxis 1			Hyperaxis 2		
w_1	w_2	w_3	w_1	w_2	w_3
-0.42	-0.04	0.54	0.49	-0.34	0.17

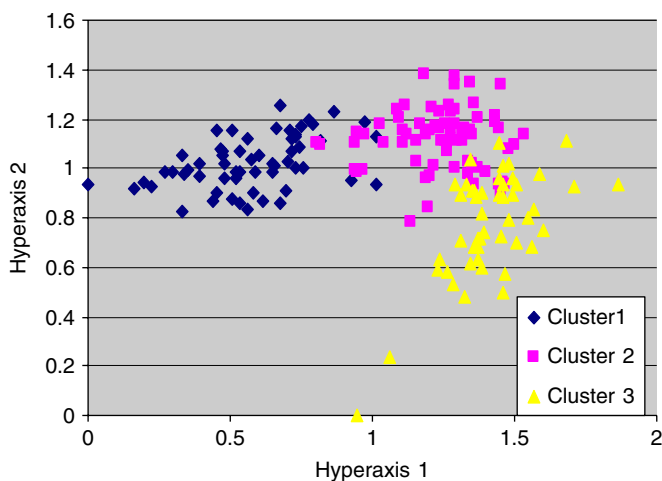


Fig. 14. A two-dimensional target space with its weight W indicated in Table 5.

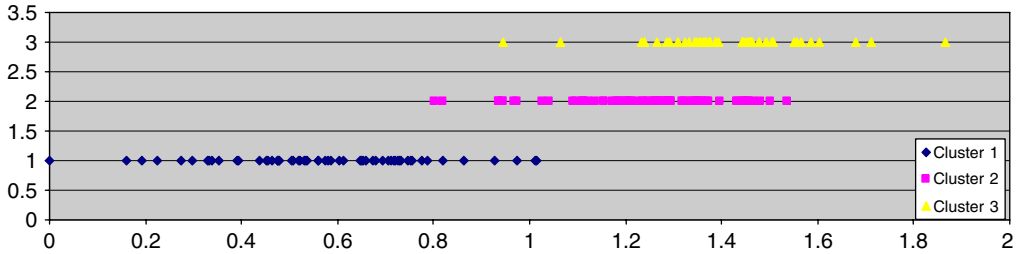


Fig. 15. Wine dataset in one-dimensional space. Cluster 3 is separated from other two clusters. The weight W is set to $(1, 0, 0, 0)$.

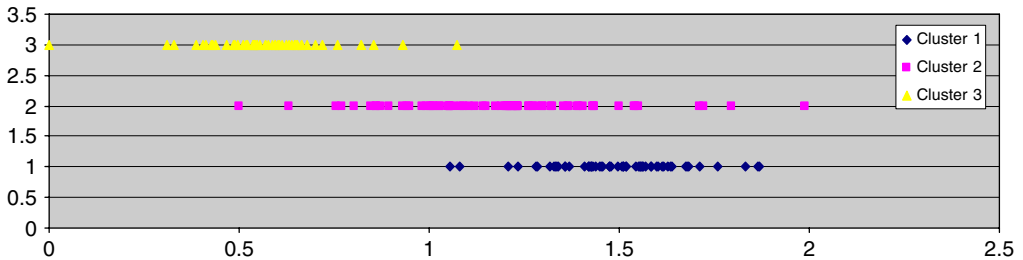


Fig. 16. Wine Dataset. Cluster 1 is separated from other two clusters. The weight W is set to $(0, 0, 0, 1)$.

We would like to discuss the stress values of the WINE dataset in more detail. From Fig. 13, we notice that selecting four or five pivot objects would generate similar stress values. We also observe the data distribution in one dimension by setting the number of pivot objects to 4. Figs. 15 and 16 show our promising results in one-dimensional target space. Cluster 3 is separated from the other two clusters with the weights set to be $W = (1, 0, 0, 0)$. Cluster 1 is also well-separated from the other two clusters with the weights of $W = (0, 0, 0, 1)$.

5. Conclusions

In this paper, we proposed an approach called *HyperMap* which is designed to map high dimensional data. From this technique, a novel concept of *hyperaxis* was introduced. A hyperaxis is a hyperplane passing through k data points, chosen by a k -center algorithm. In the specific instance when k is 2, our HyperMap is *identical* to the FastMap algorithm.

By mapping data points onto the target space spanned by the hyperaxes, hypercoordinates are obtained. Experiments on real and synthetic datasets show the utility and effectiveness of using such coordinates for both classification and visualization. HyperMap is flexible because its weights can be tuned by users.

Future research may involve clarifying the properties of weights, and searching for a more systematic method by which to determine weights. Another area of interest may be directed toward high-dimensional indexing since our approach can be used to break the “dimensionality curse”.

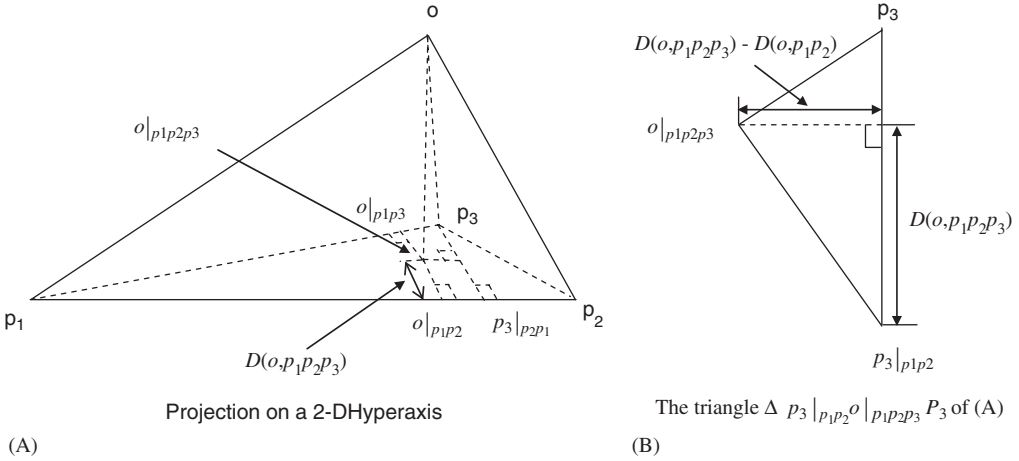


Fig. 17. Relative coordinate computing in 2-D hyperaxis.

In $\Delta p_1 p_2 p_3 \mid_{p_1 p_2}$, the following equation holds

$$(dist(p_3, p_3 \mid_{p_1 p_2}))^2 = (dist(p_1, p_3))^2 - (D(p_3, p_1 p_2))^2$$

In $\Delta o' \mid_{p_1 p_2} p_3 \mid_{p_1 p_2}$ (note $o' = o \mid_{p_1 p_2 p_3}$)

$$(dist(o', p_3 \mid_{p_1 p_2}))^2 = D^2(o, p_1 p_2 p_3) + (D(p_3, p_1 p_2) - D(o, p_1 p_2))^2$$

In $\Delta p_1 o p_3$,

$$\therefore (dist(p_1, o \mid_{p_1 p_3}))^2 - (dist(p_3, o \mid_{p_1 p_3}))^2 = (dist(o, p_1))^2 - (dist(o, p_3))^2$$

In $\Delta p_1 o' \mid_{p_1 p_3} p_3 \mid_{p_1 p_3} p_3$,

$$\begin{aligned} \therefore oo' \perp p_1 p_3 \cap oo \mid_{p_1 p_3} \perp p_1 p_3 &\Rightarrow o' o \mid_{p_1 p_3} \perp p_1 p_3 \\ \therefore (dist(o', p_3))^2 &= (dist(o', p_1))^2 - (dist(p_1, o \mid_{p_1 p_3}))^2 + (dist(p_3, o \mid_{p_1 p_3}))^2 \\ &= D^2(o, p_1 p_2 p_3) + D^2(o, p_1 p_2) - (dist(o, p_1))^2 + dist(o, p_3)^2 \end{aligned}$$

From Figure (B),
apply the cosine law in $\Delta o' p_3 p_3 \mid_{p_1 p_2}$

$$\begin{aligned} D(o, p_1 p_2 p_3) &= \frac{(dist(o', p_3 \mid_{p_1 p_2}))^2 - (dist(o', p_3))^2 + (dist(p_3, o \mid_{p_1 p_2}))^2}{2 dist(p_3, p_3 \mid_{p_1 p_2})} \\ &= \frac{(dist(o, p_1))^2 - (dist(o, p_3))^2 + (dist(p_1, p_3))^2 - 2D(p_3, p_1 p_2) \cdot D(o, p_1 p_2 p_3)}{2\sqrt{(dist(p_1, p_3))^2 - D((p_3, p_1 p_2))^2}} \end{aligned}$$

Fig. 18. HyperMap: expressions for computing relative coordinate.

Appendix A

Given an object o in a 2-D Hyperaxis determined by three pivot objects, p_1, p_2 and p_3 . Fig. 17 shows their geometrical relationship.

The object o 's first relative coordinate $D(o, p_1 p_2)$ can be computed using Eq. (5). The second relative coordinate is illustrated in Fig. 18.

References

- [1] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, *ACM Computing Surveys* 31 (3) (1999) 264–323.
- [2] I.T. Jolliffe, *Principal Component Analysis*, Springer, Berlin, 1986.
- [3] D.A. Keim, Information visualization and visual data mining, *IEEE Transactions on Visualization and Computer Graphics* 8 (1) (2002) 1–8.
- [4] D.A. Keim, H.P. Kriegel, Visdb: database exploration using multidimensional visualization, *Computer Graphics and Applications* 6 (1994) 40–49.
- [5] M. Ankerst, M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Ordering points to identify the clustering structure, *SIGMOD*, 1999, pp. 49–60.
- [6] W.S. Torgerson, *Theory and Methods of Scaling*, Wiley, New York, 1958.
- [7] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science* 290 (1) (2000) 2323–2326.
- [8] J.B. Tenenbaum, V. de. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, *Science* 290 (1) (2000) 2319–2323.
- [9] C. Faloutsos, K. I. Lin, Fastmap: a fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets, *SIGMOD*, 1995, pp. 163–174.
- [10] G. Lebanon, J. Lafferty, Hyperplane margin classifiers on the multinomial manifold, in: *Proceeding of the 21st International Conference on Machine Learning*, 2004.
- [11] K. Chen, L. Liu, VISTA: validating and refining clusters via visualization, *Information Visualization* 3 (4) (2004) 257–270.
- [12] E. Kandogan, Visualizing multi-dimensional clusters, trends, and outliers using star coordinates, *KDD*, 2001, pp. 107–116.
- [13] T. Feder, D. H. Greene, Optimal algorithms for approximate clustering, *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, 1988, pp. 434–444.
- [14] C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, J. Park, Fast algorithms for projected clustering, *SIGMOD*, 1999, pp. 61–72.
- [15] C. Aggarwal, P. Yu, Finding generalized projected clusters in high dimensional spaces, *SIGMOD*, 2000, pp. 70–81.
- [16] P. S. Bradley, O. L. Mangasarian, k -plane clustering, *Mathematical Programming Technical Report 98-08*, Department of Computer Science, University of Wisconsin-Madison, August 1998.
- [17] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an effective data clustering method for very large databases, in: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1996, pp. 103–114.