Tensor SOM and Tensor GTM: Nonlinear Tensor Analysis by Topographic Mappings

Tohru Iwasaki, Tetsuo Furukawa

Department of Human Intelligence Systems, Kyushu Institute of Technology

Email address: furukawa@brain.kyutech.ac.jp (Tetsuo Furukawa)

Tensor SOM and Tensor GTM: Nonlinear Tensor Analysis by Topographic Mappings

Tohru Iwasaki, Tetsuo Furukawa

Department of Human Intelligence Systems, Kyushu Institute of Technology

Abstract

In this paper, we propose nonlinear tensor analysis methods: the tensor self-organizing map (TSOM) and the tensor generative topographic mapping (TGTM). TSOM is a straightforward extension of the self-organizing map from high-dimensional data to tensorial data, and TGTM is an extension of the generative topographic map, which provides a theoretical background for TSOM using a probabilistic generative model. These methods are useful tools for analyzing and visualizing tensorial data, especially multimodal relational data. For given n-mode relational data, TSOM and TGTM can simultaneously organize a set of n-topographic maps. Furthermore, they can be used to explore the tensorial data space by interactively visualizing the relationships between modes. We present the TSOM algorithm and a theoretical description from the viewpoint of TGTM. Various TSOM variations and visualization techniques are also described, along with some applications to real relational datasets. Additionally, we attempt to build a comprehensive description of the TSOM family by adapting various data structures.

Keywords: self-organizing map, generative topographic map, tensor decomposition, relational data

1. Introduction

Topographic mappings are a class of dimension reduction methods that project high-dimensional data into a lower-dimensional space with preserving the topological structure. Representatives of this class include the self-organizing map (SOM) [1, 2] and the generative topographic mapping (GTM) [3, 4]. The aim of this work is to extend these methods for tensorial data, namely, to the tensor SOM (TSOM) and the tensor GTM (TGTM), which can be used to visualize multimodal relational data.

A typical application of TSOM/TGTM is the interpretation of user–item rating data, which contain product ratings for an online shop as evaluated by the users [5, 6, 7]. If the dataset consists of D-dimensional rating scores for J items evaluated by I users, then the data can be represented as an $I \times J \times D$-dimensional tensor. In such a case, we are interested in analyzing the customers, the items, and the relationships

Email address: furukawa@brain.kyutech.ac.jp (Tetsuo Furukawa)

between them. These kinds of tensorial datasets are common in many fields. Another typical example is SNS or e-mail message analysis, which can be modeled by a tensor of *(user)*×*(keyword)*×*(time)* [8, 9, 10, 11]. When recording a multi-channel electroencephalogram, the power of wavelet filters can be represented by a tensor of *(channel)*×*(frequency)*×*(time)* [12, 13]. Another possible application is image recognition for faces, postures, or gaits, because these images can be decomposed into *(person)*×*(posture)*×*(emotional expression)* [14, 15, 16, 17].

Representative methods for tensorial data analysis constitute a group of algorithms called tensor factorizations or tensor decompositions [18, 19, 20]. Tensor decompositions such as Tucker and PARAFAC involve generalizations of matrix decompositions, which decompose the given tensor into a product of lower-order matrices or tensors. Singular value decomposition (SVD) [21], principal component analysis (PCA) [22, 23], independent component analysis (ICA) [24, 25], and nonnegative matrix factorization (NMF) [26, 27] have all been extended to cope with tensorial data. The aim of this work is to add nonlinear tensor analysis tools to this group by extending the SOM and the GTM.

To analyze a multimodal relational dataset, we need to make two different types of analyses, that is, intra-mode and inter-mode analyses. For the intra-mode analysis, a low-dimensional representation is required for every mode. In the case of user–item ratings, we need two topographic maps, one for users and one for items. The user map indicates how much each user has similar or different preferences to other users, whereas the item map visualizes how much each item is preferred similarly or differently by users compared with other items. To make the inter-mode analysis, we need to visualize the relations between different modes. At times, for example, we need to analyze user preferences by focusing on an item group of interest, and at other times need to analyze preferred items by focusing on a user group of interest. For these purposes, we assume that the inherent property of each user and each item is individually represented by a low-dimensional latent variable, and that the rating score is determined by a nonlinear function of the latent variables. This latent variable model meets the framework of the GTM, a Bayesian extension of the SOM. This is the motivation for developing tensorial extensions of topographic mappings.

Another motivation of this work is to develop a simple, fast and flexible analysis tool for tensorial data. We intend for people to be able to use it easily in practical tasks, and for this reason we chose the SOM. The SOM is a popular neural network architecture with a simple algorithm. By a straightforward extension, the advantages of SOMs are inherited by TSOMs. In fact, the TSOM algorithm can be programmed with matrix multiplications only, and does not require either inverses or eigenvalues to be found. This has the advantage not only of simplicity, but also of computational cost, which is usually expensive in practical tensor data analysis. In addition, tensor data sometimes has a complex structure, and our TSOM can easily be adapted to such cases by systematically combining SOMs like building-blocks (Secs. 6 and 7). This is another advantage of using the SOM.

The remainder of this paper is organized as follows. Sec. 2 presents the theoretical background of this work and introduces some related research. Sec. 3 describes the basic TSOM and TGTM algorithms. Sec. 4 presents simulation results using artificial and real datasets, and Sec. 5 introduces the visualization techniques. Sec. 6 describes some

extensions of the TSOM. In Sec. 7, we discuss the TSOM family and the relationship between the original SOM and TSOM. Sec. 8 presents our conclusions.

2. Theoretical Preparations

2.1. Notation of vectors, matrices and tensors

An M-dimensional array of scalars is referred as a tensor of order M, each dimension of which is referred as the *mode* [18]. In the case of user–item rating data, the 1st mode corresponds to the user, and 2nd corresponds to the item.

Scalars are all in $\mathbb{R}$ and are indicated by italics, e.g., x, y. The exceptions are d, i, j, k, l, m, n and their upper cases, which are all $\mathbb{N}$. A lower case font is used for indices of vector or tensor components, and capital letters are used to represent their upper bound. Thus, $i \in \{1, \ldots, I\}$. Vectors are denoted in bold, lower case font (e.g., $\mathbf{x}$, $\mathbf{y}$), and are column vectors unless otherwise specified. Matrices are denoted in bold, upper case font (e.g., $\mathbf{X}$, $\mathbf{Y}$), and higher-order tensors are denoted by an underscore (e.g., $\underline{\mathbf{X}}$, $\underline{\mathbf{Y}}$). The (i, j) component of a matrix $\mathbf{A}$ is denoted by A_{ij}. Similarly, A_{ijk} is the (i, j, k) component of tensor $\underline{\mathbf{A}}$ of order 3. For a tensor of order M, $A_{k_1 k_2 \ldots k_M}$ is also denoted by $A_{\mathbf{k}}$, where $\mathbf{k} = (k_1, \ldots, k_M)$.

A subarray is denoted by a *colon expression*. $\mathbf{x}_{i:}$ and $\mathbf{x}_{:j}$ are the vectors consisting of the ith row and jth column components of matrix $\mathbf{X}$, respectively. Similarly, $\mathbf{x}_{ij:}$ is a vector cut out of tensor $\underline{\mathbf{X}}$ along the 3rd mode. For a tensor of order M, subtensors of order $(M-1)$ are called slices. For example, $\underline{\mathbf{A}}_{::i::}$ is the ith slice of mode 3. To describe general cases, a slice of $\underline{\mathbf{A}}$ is also denoted by $\underline{\mathbf{A}}_{i(m)}$, which means the ith slice of mode m. Thus, $\underline{\mathbf{A}}_{::i::} \equiv \underline{\mathbf{A}}_{i(3)}$.

The m-mode tensor–matrix product is denoted by $\times_m$. For example, the product of $\underline{\mathbf{X}} \in \mathbb{R}^{K_1 \times \cdots \times K_M}$ and $\mathbf{A} \in \mathbb{R}^{J \times K_m}$ becomes $\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_m \mathbf{A}$, each component of which is given by

$$Y_{k_1 \ldots k_{m-1}\, j\, k_{m+1} \ldots k_M} = \sum_{k_m=1}^{K_m} X_{k_1 \ldots k_m \ldots k_M} A_{j k_m}. \tag{1}$$

Note that $\underline{\mathbf{Y}}$ is a tensor of order M, the size of which is $K_1 \times \cdots \times K_{m-1} \times J \times K_{m+1} \times \ldots K_M$. When M matrices $\{\mathbf{A}\} = \{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}\}$ are given, the M-multiple product of $\underline{\mathbf{X}}$ and $\{\mathbf{A}\}$ is

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^{(1)} \times_2 \ldots \times_M \mathbf{A}^{(M)} = \underline{\mathbf{X}} \times \{\mathbf{A}\}. \tag{2}$$

$\underline{\mathbf{X}} \times_{-m} \{\mathbf{A}\}$ denotes the $(M-1)$-multiple product of $\underline{\mathbf{X}}$ and $\{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}\}$ except $\mathbf{A}^{(m)}$. Similarly, the multiple product of $\underline{\mathbf{X}}$ and a subset of $\{\mathbf{A}\}$ is denoted by $\underline{\mathbf{X}} \times_{\mathcal{M}'} \{\mathbf{A}\}$, where $\mathcal{M}'$ is a set of the indices of the modes. For example, $\underline{\mathbf{X}} \times_2 \mathbf{A}^{(2)} \times_4 \mathbf{A}^{(4)} \equiv \underline{\mathbf{X}} \times_{\{2,4\}} \{\mathbf{A}\}$[1].

The outer product of $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ is denoted by $\underline{\mathbf{X}} \circ \underline{\mathbf{Y}}$. For $\underline{\mathbf{X}} \in \mathbb{R}^{K_1 \times \cdots \times K_M}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, $\underline{\mathbf{Z}} = \underline{\mathbf{X}} \circ \underline{\mathbf{Y}}$ becomes $\underline{\mathbf{Z}} \in \mathbb{R}^{K_1 \times \cdots \times K_M \times J_1 \times \cdots \times J_N}$, where $Z_{\mathbf{kj}} = X_{\mathbf{k}} Y_{\mathbf{j}}$. The inner

[1] In this paper, $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ often have an extra $(M+1)$-th mode that is not multiplied by a matrix, but these notations are used in the same manner.

product is denoted by $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle$, defined by the product sum of all components. Thus $\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle \triangleq \sum_{\mathbf{k}} X_{\mathbf{k}} Y_{\mathbf{k}}$. $\left\| \underline{\mathbf{X}} \right\|$ is the Euclidean norm of $\underline{\mathbf{X}}$, defined by $\left\| \underline{\mathbf{X}} \right\| \triangleq \sqrt{\sum_{\mathbf{k}} X_{\mathbf{k}}^2} = \sqrt{\langle \underline{\mathbf{X}}, \underline{\mathbf{X}} \rangle}$.

2.2. Tensor analysis methods

A representative tensor decomposition was proposed by Tucker [28]. It decomposes a tensor into the products of a core tensor and a set of matrices [18, 19]. For a tensor of order 3, the Tucker decomposition of $\underline{\mathbf{X}}$ is

$$\underline{\mathbf{X}} \simeq \underline{\mathbf{A}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \tag{3}$$

$$= \underline{\mathbf{A}} \times \{\mathbf{U}\}. \tag{4}$$

Here, $\underline{\mathbf{A}}$ is called the core tensor. This equation means that

$$X_{k_1 k_2 k_3} \simeq \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \sum_{j_3=1}^{J_3} A_{j_1 j_2 j_3} U^{(1)}_{j_1 k_1} U^{(2)}_{j_2 k_2} U^{(3)}_{j_3 k_3}. \tag{5}$$

This type of tensor decomposition is also called Tucker3. If $\underline{\mathbf{X}}$ is decomposed into a core tensor and two matrices similar to $\underline{\mathbf{X}} \simeq \underline{\mathbf{A}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)}$, it is called Tucker2. TSOM also provides a Tucker2-like decomposition. Another representative method is PARAFAC. It decomposes a tensor into a sum of rank 1 tensors similar to $X_{k_1 k_2 k_3} = \sum_{j=1}^{J} \lambda_j R^{(1)}_{k_1 j} R^{(2)}_{k_2 j} R^{(3)}_{k_3 j}$.

Orthodox approaches for nonlinear tensorial data use the kernel method [29, 30]. An alternative approach is to use manifold learning methods [31, 32]. The method most related to our study is SOM^2, which is also referred as 'SOM of SOMs' [33, 34]. SOM^2 visualizes the relationships between a set of datasets. To achieve this, SOM^2 has a hierarchical architecture; the lower level generates a set of models for the given datasets, and the higher level generates a map of lower models. Theoretically, SOM^2 models a set of datasets using a geometrical structure called a *fiber bundle*, which is also represented by a tensor. The essential difference between TSOM and SOM^2 is in their data structures. From our viewpoint, both algorithms can be unified into the same family of algorithms, which can be adapted to various data structures. We discuss this issue in Sec. 7.

2.3. Self-organizing map: SOM

We first briefly summarize SOM as an introduction to TSOM. Suppose that Ω is the universe of the objects of interest, and a data vector $\mathbf{x}(\omega) \in \mathcal{X}$ is observed from a sample point $\omega \in \Omega$. Here, $\mathcal{X}$ is the observation space, which is assumed to be $\mathcal{X} = \mathbb{R}^D$ in this paper. Let $\Omega_S = \{\omega_1, \ldots, \omega_N\}$ be a sample set from the universe Ω, and $\mathbf{x}_{n:}$ be the data vector observed from ω_n. Thus, $\mathbf{X} = (\mathbf{x}_{n:})$ becomes a $N \times D$ data matrix. From the viewpoint of a generative model, $\mathbf{x}_{n:}$ is assumed to be generated from the latent variable $\mathbf{z}_{n:} \in \mathcal{Z}$ and a nonlinear smooth map $f(\mathbf{z})$, so that $\mathbf{x}_{n:} = f(\mathbf{z}_{n:}) + \boldsymbol{\varepsilon}$. Here, $\boldsymbol{\varepsilon}$ is isotropic Gaussian noise. The task of the SOM is to estimate the latent variables $(\mathbf{z}_{n:})$ and the nonlinear map $f(\mathbf{z})$, so that they represent the observed data by

$\mathbf{x}_{n:} \simeq f(\mathbf{z}_{n:})$. As the result, we obtain a topographic map of the objects in the latent space $\mathfrak{Z}$. Usually $\mathfrak{Z}$ is assumed to be a closed area in a low dimensional space $\mathbb{R}^L$, typically the 2-dimensional square space. In this paper, the prior of $\mathbf{z}$ is the uniform distribution in $\mathfrak{Z} = [-1, 1]^L \subset \mathbb{R}^L$ ($L = 1$ or 2), but it is easily generalized to other cases.

To represent the nonlinear map $f(\mathbf{z})$, we discretize the latent space $\mathfrak{Z}$ into K regular nodes. Let $\boldsymbol{\zeta}_{k:}$ be the positional vector of the kth node in $\mathfrak{Z}$, and $\mathbf{y}_{k:} \triangleq f(\boldsymbol{\zeta}_{k:})$. Then the matrix $\mathbf{Y} = (\mathbf{y}_{k:})$ represents the entire map. Because $\mathbf{y}_{k:}$ indicates the corresponding point in $\mathfrak{X}$ for the kth node in $\mathfrak{Z}$, it is often referred to as the 'reference vector'.

The SOM algorithm is an expectation-maximization (EM) algorithm in a broad sense, in which we iterate over the expectation (E) and maximization (M) steps while reducing the neighborhood area until the map converges [35, 36]. Considering the extension to tensor cases, let us summarize the SOM algorithm using matrix notation.

In the E step, the maximum a posteriori node is the winning node for each data vector. That is,

$$k_n^* = \arg\min_k \|\mathbf{x}_{n:} - \mathbf{y}_{k:}\|^2 , \tag{6}$$

$$B_{kn} = \delta(k, k_n^*), \tag{7}$$

where $\mathbf{B} = (B_{kn}) \in \mathbb{R}^{K\times N}$ is the winning matrix, and δ is Kronecker's delta. Let $\mathbf{H} \in \mathbb{R}^{K\times K}$ be the neighborhood matrix given by

$$H_{kk'} = \exp\left[-\frac{1}{2\sigma^2(t)} \left\|\boldsymbol{\zeta}_{k:} - \boldsymbol{\zeta}_{k':}\right\|^2\right]. \tag{8}$$

$\sigma(t)$ is the neighborhood size, which is gradually reduced with learning time t. This neighborhood shrinkage is necessary for avoiding local minima, as in simulated annealing. Using the winning matrix $\mathbf{B}$ and the neighborhood matrix $\mathbf{H}$, the responsibility matrix $\mathbf{R} \in \mathbb{R}^{K\times N}$ is determined by

$$\mathbf{R} = \mathbf{HB}. \tag{9}$$

The following two matrices are also calculated for the M step.

$$\mathbf{G} = \mathrm{diag}\left(\sum_{n=1}^{N} R_{kn}\right) \tag{10}$$

$$\tilde{\mathbf{R}} = \mathbf{G}^{-1}\mathbf{R}, \tag{11}$$

where $\mathbf{G} \in \mathbb{R}^{K\times K}$ is the diagonal matrix, which represents the sum of the responsibility of each node, and $\tilde{\mathbf{R}} \in \mathbb{R}^{K\times N}$ is the responsibility normalized by $\mathbf{G}$.

In the M step, the map $\mathbf{Y}$ is updated so that the expectation of the square error is minimized using

$$\mathbf{y}_{k:} = \frac{1}{G_k} \sum_{n=1}^{N} R_{kn}\mathbf{x}_{n:}. \tag{12}$$

In matrix notation, (12) is

$$\mathbf{Y} = \tilde{\mathbf{R}}\mathbf{X}. \tag{13}$$

Using the tensor–matrix product notation, (13) can be written as

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}, \tag{14}$$

where the data matrix $\mathbf{X}$ and the map matrix $\mathbf{Y}$ are regarded as tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ of order 2. We iterate over these steps, reducing the neighborhood size until the map converges.

Many previous studies have shown that the objective function of the SOM is

$$F = -\frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{K} H_{k\,k_n^*} \|\mathbf{x}_{n:} - \mathbf{y}_{k:}\|^2 \tag{15}$$

[35, 36, 37, 38, 39]. We can easily derive the M step (13) using this objective function, and slightly modify the E step (6) to

$$k_n^* = \arg\min_{k'} \sum_{k=1}^{K} H_{k\,k'} \|\mathbf{x}_{n:} - \mathbf{y}_{k:}\|^2 . \tag{16}$$

The original SOM algorithm (6) can be regarded as an approximation of (16), because they are almost equal when the neighborhood is sufficiently small. Note that F is a function of σ, which gradually changes throughout the learning process.

2.4. SOM with basis functions

It is also possible to represent the nonlinear map $f(\mathbf{z})$ by a linear combination of basis functions, instead of a discrete number of reference vectors. Basis functions have the advantage of allowing us to deal with continuous, differentiable maps with arbitrary resolutions. Now, suppose that $\{\varphi_1(\mathbf{z}), \ldots, \varphi_J(\mathbf{z})\}$ is an appropriate basis set defined on $\mathfrak{Z}$, and the nonlinear map $f(\mathbf{z})$ can be represented by

$$f(\mathbf{z}) \simeq \sum_{j=1}^{J} \mathbf{w}_j \varphi_j(\mathbf{z}) = \mathbf{W}\boldsymbol{\varphi}(\mathbf{z}) \tag{17}$$

with sufficient accuracy. In this case, $\mathbf{W}$ is updated using

$$\begin{aligned}\mathbf{W} &= (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{G}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{X} \\ &= \boldsymbol{\Phi}_G^{\#}\tilde{\mathbf{R}}\mathbf{X},\end{aligned} \tag{18}$$

so that it minimizes the expectation error. Here, $\boldsymbol{\Phi} \triangleq \left(\varphi_j(\boldsymbol{\zeta}_{k:})\right) \in \mathbb{R}^{K\times J}$ and $\boldsymbol{\Phi}_G^{\#}$ is the generalized inverse of $\boldsymbol{\Phi}$ with the weight $\mathbf{G}$. Thus, $\boldsymbol{\Phi}_G^{\#} \triangleq (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{G}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{G}$. Defining the $J \times N$ matrix $\tilde{\mathbf{Q}}$ by

$$\tilde{\mathbf{Q}} \triangleq \boldsymbol{\Phi}_G^{\#}\tilde{\mathbf{R}} = (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{G}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}, \tag{19}$$

(18) becomes

$$\mathbf{W} = \tilde{\mathbf{Q}}\mathbf{X}. \tag{20}$$

The difference between (13) and (20) is that $\mathbf{Y}$ and $\tilde{\mathbf{R}}$ are replaced by $\mathbf{W}$ and $\tilde{\mathbf{Q}}$.

To update $\mathbf{W}$ using (18), $\mathbf{\Phi}_G^{\#}$ should be calculated at every iteration. Because G_k is the sum of the responsibilities, G_k is more or less equal to NP_k, where P_k is the prior of the kth node. Therefore, $\mathbf{\Phi}_G^{\#}$ can be approximated by $\mathbf{\Phi}_P^{\#}$, which we can calculate in advance. In the case of the uniform prior, $\mathbf{\Phi}_P^{\#}$ becomes the ordinary Moore-Penrose inverse $\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}$.

Another advantage of using basis functions is that gradient methods can be used for the E step. In the case of the gradient descent method, the latent variable $\mathbf{z}_{n:} = (z_{nl})^{\mathrm{T}} \in \mathbb{R}^L$ is updated by

$$z_{nl} := z_{nl} - \eta \left\langle \mathbf{W}\boldsymbol{\varphi}(\mathbf{z}_{n:}) - \mathbf{x}_n, \mathbf{W}\boldsymbol{\varphi}_l'(\mathbf{z}_{n:}) \right\rangle, \tag{21}$$

where $\boldsymbol{\varphi}_l'(\mathbf{z}) \triangleq \partial\boldsymbol{\varphi}(\mathbf{z})/\partial z_l$. In this case, $\mathbf{R}$ is calculated by

$$R_{kn} = \exp\left[-\frac{1}{2\sigma^2(t)}\left\|\boldsymbol{\zeta}_{k:} - \mathbf{z}_{n:}\right\|^2\right]. \tag{22}$$

Note that a single step of the gradient descent method is enough for every E step, because the convergence speed is limited by the neighborhood size scheduling. Thus there is no need to use other faster methods for this purpose.

2.5. Generative topographic mapping: GTM

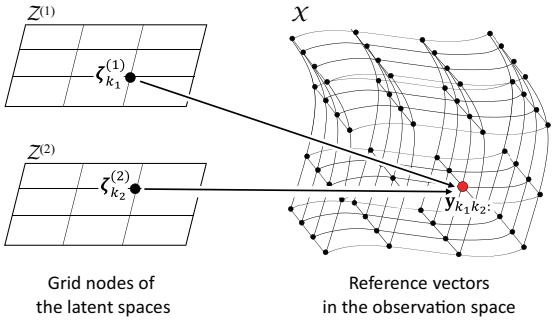
The generative topographic mapping (GTM) is a SOM-like algorithm, which is derived from a probabilistic generative model [3, 4, 40]. Thus GTM can be regarded as a theoretical model for SOM. The GTM algorithm can be described by the EM algorithm (or the variational Bayesian method). In the E step, the responsibility is calculated using

$$R_{kn} = \frac{\exp\left[-\frac{\beta}{2}\left\|\mathbf{x}_{n:} - \mathbf{y}_{k:}\right\|^2\right]}{\sum_{k'=1}^{K}\exp\left[-\frac{\beta}{2}\left\|\mathbf{x}_{n:} - \mathbf{y}_{k':}\right\|^2\right]}. \tag{23}$$

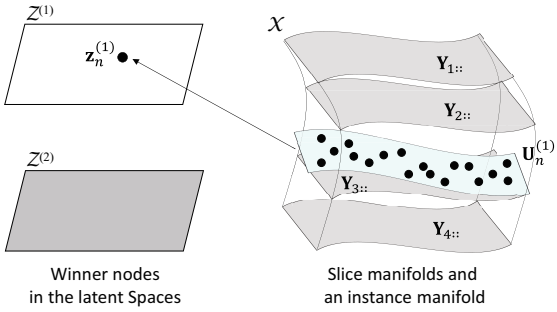
Unlike SOM, GTM does not have a neighborhood relationship in the E step. Instead, GTM introduces other assumptions in the M step, to obtain a smooth continuous map. Thus, the original GTM uses Gaussian radial basis functions [3], and another type of GTM represents the map using a Gaussian stochastic process [4, 40]. In the former case, the M step is

$$\mathbf{W} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{G}\mathbf{\Phi} + \frac{\lambda}{\beta}\mathbf{I}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{X}. \tag{24}$$

Here, $\mathbf{\Phi}$ is the matrix of the radial bases and the term $\frac{\lambda}{\beta}\mathbf{I}$ represents the regularization. Letting $\tilde{\mathbf{Q}} \triangleq \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{G}\mathbf{\Phi} + \frac{\lambda}{\beta}\mathbf{I}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}$, we can represent the M step by $\mathbf{W} = \tilde{\mathbf{Q}}\mathbf{X}$, as in



(a)



(b)

Figure 1: (a) The architecture of TSOM–R^2. The map from the latent spaces to the observation space is represented by the set of reference vectors ($\mathbf{y}_{k_1 k_2:}$), each of which corresponds to a pair of nodes ($\zeta^{(1)}_{k_1}, \zeta^{(2)}_{k_2}$) in the latent spaces. The reference vectors represent a product manifold in the observation space. (Note that the product manifold is 4-dimensional in this case, and is depicted as a 3-dimensional nonlinear cube in this figure). (b) Estimates of the instance manifold $\{\mathbf{U}^{(m)}_n\}$ for each instance $\omega^{(m)}_n$. The winner is determined according to the best matching slice manifold of the instance manifold.

(20). In the latter case, the M step is

$$\mathbf{Y} = \left(\mathbf{G} + \frac{\lambda}{\beta}\mathbf{H}^{-1}\right)^{-1} \mathbf{R}\mathbf{X}, \tag{25}$$

where the neighborhood matrix $\mathbf{H}$ is used as the correlation matrix for the Gaussian process. Note that both algorithms are equivalent if the radial bases satisfy $\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{H}$.

3. Basic TSOM

3.1. Definition of the task

To clarify the task of the TSOM, let us revisit the example of the user–item rating. In this case, the objects of interest are the set of users $\Omega^{(\mathrm{user})}$, and the set of items for sale $\Omega^{(\mathrm{item})}$. Because each rating score $\mathbf{x} \in \mathbb{R}^D$ is determined by a pair of instances $(\omega^{(\mathrm{user})}, \omega^{(\mathrm{item})})$, we can call this type of data 2-mode relational data. The data tensor

$\underline{\mathbf{X}}$ is of order 3, and its modes correspond to the users, items, and rating components. Suppose that the last mode is not the target of the tensor analysis, similar to Tucker2. (If necessary, the data can be regarded as 3-mode relational data, which can be also dealt with by a TSOM).

The aim of the TSOM is to visualize the relationships between objects within each mode (i.e., between users and between items), and simultaneously visualize the relationships between two modes (i.e., between users and items). In this paper, the TSOM for 2-mode relational data is abbreviated to TSOM–R^2.

This situation can be generalized as follows. Suppose that the set of object universes to be analyzed is $\left\{\Omega^{(1)}, \ldots, \Omega^{(M)}\right\}$, and $\Omega_S^{(m)} = \left\{\omega_1^{(m)}, \ldots, \omega_{N_m}^{(m)}\right\}$ is the sample set of $\Omega^{(m)}$, consisting of N_m instances. We observe the D-dimensional data $\mathbf{x}_{n_1 \ldots n_M:} = \mathbf{x}\left(\omega_{n_1}^{(1)}, \ldots, \omega_{n_M}^{(M)}\right)$ for all members of the direct product set $\Omega_S = \prod_{m=1}^{M} \Omega_S^{(m)}$. Then, we obtain M-mode relational data $\underline{\mathbf{X}}$, which is a tensor of order $(M+1)$.

Given such tensorial data, TSOM–R^M must organize M topographic maps (each of which visualizes the relationships of objects *within each mode*) and visualize the relationships *between modes*. TSOM–R^M assumes that there are M latent spaces $\{\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(M)}\}$, and that the observed data can be represented by a nonlinear map f of M latent variables. That is,

$$\mathbf{x}_{n_1 \ldots n_M:} \simeq f\left(\mathbf{z}_{n_1:}^{(1)}, \ldots, \mathbf{z}_{n_M:}^{(M)}\right). \tag{26}$$

Thus, the actual TSOM task is to simultaneously estimate the nonlinear map and the latent variables.

For ordinary SOM, the dimensions of the observation space (D) and of the latent space (L) should satisfy $D \geq L$, so that the topology is preserved. Note that this restriction is not necessary for TSOM. In fact, we can even consider a scalar case (i.e., $D = 1$). This is because we expect topological preservation between $\left\{\underline{\mathbf{X}}_{n(m)}\right\}$ and $\left\{\mathbf{z}_{n:}^{(m)}\right\}$, and the dimension of $\underline{\mathbf{X}}_{n(m)}$ is usually much greater than L.

3.2. TSOM–R^2 algorithm

We first present the TSOM algorithm for 2-mode relational data, namely, TSOM–R^2. In this case, the relational dataset is given as a tensor of order 3, $\underline{\mathbf{X}} = (x_{n_1 n_2 d})$. The aim of TSOM–R^2 is to model the 2-mode relational data using a nonlinear map. That is,

$$\begin{aligned} f : \mathcal{Z}^{(1)} \times \mathcal{Z}^{(2)} &\longrightarrow \mathcal{Y} \subset \mathcal{X} = \mathbb{R}^D \\ (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) &\longmapsto \mathbf{y}(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}). \end{aligned} \tag{27}$$

To represent f, TSOM–R^2 has two latent spaces, $\mathcal{Z}^{(1)}$ and $\mathcal{Z}^{(2)}$, each of which is discretized to K_1 and K_2 nodes, respectively. Letting $\boldsymbol{\zeta}_{k_m}^{(m)}$ be the coordinate of the k_mth node of the mth latent space, we assign a reference vector to every pair of nodes (k_1, k_2), i.e., $\mathbf{y}_{k_1 k_2:} = f(\boldsymbol{\zeta}_{k_1}^{(1)}, \boldsymbol{\zeta}_{k_2}^{(2)})$. Consequently, the entire map is represented by the tensor $\underline{\mathbf{Y}} = (Y_{k_1 k_2 d})$, as shown in Fig. 1 (a).

In the TSOM–R^2 algorithm, *slice manifolds* and *instance manifolds* are essential for determining the winning nodes. The slice manifolds are the submanifolds of $\mathcal{Y}$,

which are represented by the slices of $\underline{\mathbf{Y}}$. For example, the k_1th slice manifold of mode 1 is defined by $\mathcal{Y}_{k_1}^{(1)} = f\left(\boldsymbol{\zeta}_{k_1}^{(1)}, \mathcal{Z}^{(2)}\right)$, which is represented by $\mathbf{Y}_{k_1::}$. The slice manifold characterizes the data distribution for a specified latent variable. In contrast, the instance manifold characterizes the data distribution when an instance is specified. Now, suppose that an instance $\omega_{n_1}^{(1)}$ is specified, and the data distribution $\{\mathbf{x}_{n_1 1}, \ldots, \mathbf{x}_{n_1 N_2}\}$ is approximated by $\mathbf{x}_{n_1 n_2} \simeq f^{(1)}\left(\mathbf{z}_{n_2:}^{(2)} \,\middle|\, \omega_{n_1}^{(1)}\right)$. Then, the instance manifold $\mathcal{U}_{n_1}^{(1)}$ is given by $\mathcal{U}_{n_1}^{(1)} = f^{(1)}\left(\mathcal{Z}^{(2)} \,\middle|\, \omega_{n_1}^{(1)}\right)$. Suppose that this manifold is represented by a matrix $\mathbf{U}_{n_1::}^{(1)}$, defined by $\mathbf{u}_{n_1 k_2:}^{(1)} = f^{(1)}\left(\boldsymbol{\zeta}_{k_2}^{(2)} \,\middle|\, \omega_{n_1}^{(1)}\right)$. Thus, the instance manifold $\mathbf{U}_{n_1::}^{(1)}$ is the n_1th slice of $\underline{\mathbf{U}}^{(1)}$. The instance manifolds of mode 2 are also defined in the same way. In the TSOM algorithm, the instance manifolds are estimated for all instances, and they are regarded as the feature vectors for determining the winners. This situation is shown in Fig. 1 (b).

Like the ordinary SOM, the TSOM algorithm consists of an E step and an M step. For unfamiliar readers, we denote the algorithm in two ways, with and without the tensor convention.

E step

In the E step, the winner of each mode is determined by

$$k_{n_1}^{*(1)} = \arg\min_{k_1} \left\| \mathbf{Y}_{k_1::} - \mathbf{U}_{n_1::}^{(1)} \right\|^2 \tag{28}$$

$$= \arg\min_{k_1} \sum_{k_2=1}^{K_2} \sum_{d=1}^{D} \left(Y_{k_1 k_2 d} - U_{n_1 k_2 d}^{(1)} \right)^2 \tag{29}$$

$$k_{n_2}^{*(2)} = \arg\min_{k_2} \left\| \mathbf{Y}_{:k_2:} - \mathbf{U}_{:n_2:}^{(2)} \right\|^2 \tag{30}$$

$$= \arg\min_{k_2} \sum_{k_1=1}^{K_1} \sum_{d=1}^{D} \left(Y_{k_1 k_2 d} - U_{k_1 n_2 d}^{(2)} \right)^2 \tag{31}$$

Thus, the winner of $\omega_{n_1}^{(1)}$ is determined as the best matching slice manifold for the instance manifold $\mathbf{U}_{n_1::}$ (Fig. 1 (b)). (28) also means that $\mathbf{U}_{n_1::}^{(1)}$ and $\mathbf{Y}_{k_1::}$ behave as if they are the data and the reference vectors with respect to mode 1.

After determining all the winners, the winning matrices $\mathbf{B}^{(1)}$ and $\mathbf{B}^{(2)}$, and the responsibility matrices $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ are obtained using

$$B_{k_1 n_1}^{(1)} = \delta(k_1, k_{n_1}^{*(1)}) \tag{32}$$

$$B_{k_2 n_2}^{(2)} = \delta(k_2, k_{n_2}^{*(2)}) \tag{33}$$

$$\mathbf{R}^{(1)} = \mathbf{H}^{(1)} \mathbf{B}^{(1)} \tag{34}$$

$$\mathbf{R}^{(2)} = \mathbf{H}^{(2)} \mathbf{B}^{(2)}. \tag{35}$$

Here, $\mathbf{H}^{(m)}$ is the neighborhood matrix of mode m. Elementwise, these equations can

be represented by

$$R^{(1)}_{k_1 n_1} = \exp\left[-\frac{1}{{\sigma^{(1)}}^2} d^2(k_1, k^{*(1)}_{n_1})\right] \tag{36}$$

$$R^{(2)}_{k_2 n_2} = \exp\left[-\frac{1}{{\sigma^{(2)}}^2} d^2(k_2, k^{*(2)}_{n_2})\right], \tag{37}$$

where $d(k, k')$ is the distance between two nodes in the latent space. Similar to (10) and (11), the sum of the responsibility $\mathbf{G}^{(m)}$ and normalized responsibility $\tilde{\mathbf{R}}^{(m)}$ are also calculated for each mode using

$$\mathbf{G}^{(1)} = \mathrm{diag}\left(\sum_{n_1=1}^{N_1} R^{(1)}_{k_1 n_1}\right) \tag{38}$$

$$\mathbf{G}^{(2)} = \mathrm{diag}\left(\sum_{n_2=1}^{N_2} R^{(2)}_{k_2 n_2}\right) \tag{39}$$

$$\tilde{\mathbf{R}}^{(1)} = {\mathbf{G}^{(1)}}^{-1}\mathbf{R}^{(1)} \tag{40}$$

$$\tilde{\mathbf{R}}^{(2)} = {\mathbf{G}^{(2)}}^{-1}\mathbf{R}^{(2)}. \tag{41}$$

M step

In the M step, tensors $\underline{\mathbf{Y}}$, $\underline{\mathbf{U}}^{(1)}$, and $\underline{\mathbf{U}}^{(2)}$ are updated using

$$\underline{\mathbf{U}}^{(1)} = \underline{\mathbf{X}} \times_2 \tilde{\mathbf{R}}^{(2)} \tag{42}$$

$$\underline{\mathbf{U}}^{(2)} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \tag{43}$$

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_2 \tilde{\mathbf{R}}^{(2)}. \tag{44}$$

If the components of the tensors are denoted explicitly, the above equations can be rewritten as

$$U^{(1)}_{n_1 k_2 d} = \frac{1}{G^{(2)}_{k_2}} \sum_{n_2=1}^{N_2} R^{(2)}_{k_2 n_2} X_{n_1 n_2 d} \tag{45}$$

$$U^{(2)}_{k_1 n_2 d} = \frac{1}{G^{(1)}_{k_1}} \sum_{n_1=1}^{N_1} R^{(1)}_{k_1 n_1} X_{n_1 n_2 d} \tag{46}$$

$$Y_{k_1 k_2 d} = \frac{1}{G^{(1)}_{k_1} G^{(2)}_{k_2}} \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} R^{(1)}_{k_1 n_1} R^{(2)}_{k_2 n_2} X_{n_1 n_2 d}. \tag{47}$$

Note that (44) is also denoted by

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}^{(1)} \times_1 \tilde{\mathbf{R}}^{(1)} \tag{48}$$

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}^{(2)} \times_2 \tilde{\mathbf{R}}^{(2)}. \tag{49}$$

Computationally, it is quicker to evaluate $\underline{\mathbf{Y}}$ by (48) or (49) than (44).

3.3. TSOM–R^M algorithm

We now present the generalized TSOM algorithm for the arbitrary M-mode relational data, namely, TSOM–R^M. In this case, the data tensor $\underline{\mathbf{X}}$ and the map tensor $\underline{\mathbf{Y}}$ become tensors of order $(M+1)$, which are $(N_1 \times \cdots \times N_M \times D)$ and $(K_1 \times \cdots \times K_M \times D)$ in size, respectively.

E step

As in TSOM–R^2, the winners are determined by the distance between the instance manifolds $\underline{\mathbf{U}}^{(m)}_{n(m)}$ and the slice manifolds $\underline{\mathbf{Y}}_{k(m)}$. That is,

$$k_n^{*(m)} = \arg\min_k \left\| \underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{U}}^{(m)}_{n(m)} \right\|^2 \tag{50}$$

$$B^{(m)}_{kn} = \delta\left(k, k_n^{*(m)}\right). \tag{51}$$

Then, the matrices $\mathbf{R}^{(m)}$, $\mathbf{G}^{(m)}$, and $\tilde{\mathbf{R}}^{(m)}$ are calculated for each mode using

$$\mathbf{R}^{(m)} = \mathbf{H}^{(m)}\mathbf{B}^{(m)} \tag{52}$$

$$\mathbf{G}^{(m)} = \mathrm{diag}\left(\sum_{n=1}^{N_m} R^{(m)}_{kn}\right) \tag{53}$$

$$\tilde{\mathbf{R}}^{(m)} = {\mathbf{G}^{(m)}}^{-1}\mathbf{R}^{(m)}. \tag{54}$$

M step

In the M step, $\underline{\mathbf{Y}}$ and $\{\underline{\mathbf{U}}^{(m)}\}$ are updated according to

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times \left\{\tilde{\mathbf{R}}\right\} \tag{55}$$

$$\underline{\mathbf{U}}^{(m)} = \underline{\mathbf{X}} \times_{-m} \left\{\tilde{\mathbf{R}}\right\}. \tag{56}$$

We iterate over these two steps, reducing the neighborhood size until the map converges.

Consider the algorithm for calculating (55) and (56). If these tensors are calculated independently, we require M^2 tensor–matrix multiplications. For example, if $M = 4$, (55) and (56) become

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_2 \tilde{\mathbf{R}}^{(2)} \times_3 \tilde{\mathbf{R}}^{(3)} \times_4 \tilde{\mathbf{R}}^{(4)} \tag{57}$$

$$\underline{\mathbf{U}}^{(1)} = \underline{\mathbf{X}} \times_2 \tilde{\mathbf{R}}^{(2)} \times_3 \tilde{\mathbf{R}}^{(3)} \times_4 \tilde{\mathbf{R}}^{(4)} \tag{58}$$

$$\underline{\mathbf{U}}^{(2)} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_3 \tilde{\mathbf{R}}^{(3)} \times_4 \tilde{\mathbf{R}}^{(4)} \tag{59}$$

$$\underline{\mathbf{U}}^{(3)} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_2 \tilde{\mathbf{R}}^{(2)} \times_4 \tilde{\mathbf{R}}^{(4)} \tag{60}$$

$$\underline{\mathbf{U}}^{(4)} = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_2 \tilde{\mathbf{R}}^{(2)} \times_3 \tilde{\mathbf{R}}^{(3)}, \tag{61}$$

in which there are 16 tensor–matrix multiplications. However, we can reduce this to $\lceil M \log_2 M + 1 \rceil$ multiplications. Suppose that $\mathcal{M} = \{1, \ldots, M\}$ is a set of modes, and

$\mathcal{M}'$ is a subset of $\mathcal{M}$. Suppose further that the intermediate product tensor $\underline{\mathbf{P}}(\mathcal{M}')$ is defined by

$$\underline{\mathbf{P}}(\mathcal{M}') \triangleq \underline{\mathbf{X}} \times_{\mathcal{M}'} \left\{\tilde{\mathbf{R}}\right\}. \tag{62}$$

Note that $\underline{\mathbf{P}}(\mathcal{M}' + \{m\}) = \underline{\mathbf{P}}(\mathcal{M}') \times_m \tilde{\mathbf{R}}^{(m)}$, if $m \notin \mathcal{M}'$. Considering $\underline{\mathbf{Y}} = \underline{\mathbf{P}}(\mathcal{M})$ and $\underline{\mathbf{U}}^{(m)} = \underline{\mathbf{P}}(\mathcal{M} \setminus \{m\})$, (57)–(61) can be calculated using

$$\underline{\mathbf{P}}(\{1,2\}) = \underline{\mathbf{X}} \times_1 \tilde{\mathbf{R}}^{(1)} \times_2 \tilde{\mathbf{R}}^{(2)} \tag{63}$$

$$\underline{\mathbf{P}}(\{3,4\}) = \underline{\mathbf{X}} \times_3 \tilde{\mathbf{R}}^{(3)} \times_4 \tilde{\mathbf{R}}^{(4)} \tag{64}$$

$$\underline{\mathbf{U}}^{(1)} = \underline{\mathbf{P}}(\{3,4\}) \times_2 \tilde{\mathbf{R}}^{(2)} \tag{65}$$

$$\underline{\mathbf{U}}^{(2)} = \underline{\mathbf{P}}(\{3,4\}) \times_1 \tilde{\mathbf{R}}^{(1)} \tag{66}$$

$$\underline{\mathbf{U}}^{(3)} = \underline{\mathbf{P}}(\{1,2\}) \times_4 \tilde{\mathbf{R}}^{(4)} \tag{67}$$

$$\underline{\mathbf{U}}^{(4)} = \underline{\mathbf{P}}(\{1,2\}) \times_3 \tilde{\mathbf{R}}^{(3)} \tag{68}$$

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}^{(1)} \times_1 \tilde{\mathbf{R}}^{(1)}. \tag{69}$$

Thus, these tensors can be calculated in parallel like a binary-tree algorithm. In this case, the number of tensor–matrix multiplications is reduced from 16 to 9 ($= 4\log_2 4 + 1$).

3.4. Derivation of the TSOM algorithm

In this subsection, we present a theoretical derivation of the TSOM algorithm. To begin with, let us start from the notation of the weighted square error between two tensors. Suppose that $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ are tensors of order $(M+1)$, and $\{\mathbf{R}\} = \{\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(M)}\}$ is a set of responsibility matrices that give the weights. The weighted error between $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ is defined as

$$E\left(\underline{\mathbf{X}}, \underline{\mathbf{Y}}; \{\mathbf{R}\}\right) \triangleq \sum_{n_1,\ldots,n_M} \sum_{k_1,\ldots,k_M} \left(\prod_{m=1}^{M} R^{(m)}_{k_m n_m}\right) \left\|\mathbf{x}_{n_1 \ldots n_M :} - \mathbf{y}_{k_1 \ldots k_M :}\right\|^2. \tag{70}$$

If each $\mathbf{R}^{(m)}$ is normalized so that $\sum_k R^{(m)}_{kn} = 1$, (70) gives the expectation error. Let the square errors of two tensors except mode m be defined as

$$E^{(-m)}_{k_m n_m}(\underline{\mathbf{X}}, \underline{\mathbf{Y}}; \{\mathbf{R}\}) \triangleq \sum_{\substack{n_1,\ldots,n_M \\ \text{except } n_m}} \sum_{\substack{k_1,\ldots,k_M \\ \text{except } k_m}} \left(\prod_{\substack{m'=1 \\ \text{except } m}}^{M} R^{(m')}_{k_{m'} n_{m'}}\right) \left\|\mathbf{x}_{n_1 \ldots n_M :} - \mathbf{y}_{k_1 \ldots k_M :}\right\|^2. \tag{71}$$

Note that $\mathbf{E}^{(-m)}(\underline{\mathbf{X}}, \underline{\mathbf{Y}}; \{\mathbf{R}\})$ becomes a $K_m \times N_m$ matrix.

Using this weighted square error, the objective function of TSOM can be defined as

$$F \triangleq -\frac{1}{2N} E\left(\underline{\mathbf{X}}, \underline{\mathbf{Y}}; \{\mathbf{R}\}\right). \tag{72}$$

Here, $N = \prod_m N_m$ and $R^{(m)}_{kn} = H^{(m)}_{k\,k^{*(m)}_n}$. This is a straightforward extension of (15). Note that (72) can be decomposed with respect to mode m, by applying Pythagoras' theorem. That is,

$$F = -\frac{1}{2N_m}\sum_{n=1}^{N_m}\sum_{k=1}^{K_m} R^{(m)}_{kn}\left\{\frac{1}{K_{-m}}\left\|\underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{U}}^{(m)}_{n(m)}\right\|^2 + \frac{1}{N_{-m}}E^{(-m)}_{kn}\left(\underline{\mathbf{X}}, \underline{\mathbf{U}}^{(m)}; \{\mathbf{R}\}\right)\right\}, \quad (73)$$

where $N_{-m} = \prod_{m'\neq m} N_{m'}$, $K_{-m} = \prod_{m'\neq m} K_{m'}$, and $\underline{\mathbf{U}}^{(m)}$ is the instance manifold given by (56).

In the E step, (72) is maximized with respect to $\{k^{*(m)}_n\}$ according to

$$k^{*(m)}_n = \arg\min_k \sum_{k'} H^{(m)}_{kk'}\, E^{(-m)}_{k'n_m}(\underline{\mathbf{X}}, \underline{\mathbf{Y}};\, \{\mathbf{R}\}). \quad (74)$$

Applying (73), (74) becomes

$$k^{*(m)}_n = \arg\min_k \sum_{k'} H^{(m)}_{kk'}\left\|\underline{\mathbf{Y}}_{k'(m)} - \underline{\mathbf{U}}^{(m)}_{n(m)}\right\|^2, \quad (75)$$

because the second term of (73) is independent of $k^{*(m)}_n$. Similar to the approximations from (16) to (6), (74) and (75) become

$$k^{*(m)}_n = \arg\min_k E^{(-m)}_{kn}(\underline{\mathbf{X}}, \underline{\mathbf{Y}};\, \{\mathbf{R}\}) \quad (76)$$

$$= \arg\min_k \left\|\underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{U}}^{(m)}_{n(m)}\right\|^2. \quad (77)$$

Thus, we obtain (50). Both (76) and (77) yield the same result, but (77) can be calculated quicker than (76).

In the M step, (72) is maximized with respect to $\underline{\mathbf{Y}}$. It is easy to show that the solution is given by (55).

3.5. TSOM with basis functions

If the order of the tensor increases, the computational costs drastically increase. The size of the task is roughly proportional to K^M. To reduce the costs, we need to reduce the task size as well as algorithmic improvement. The easiest way is to reduce K, i.e., the node number. However, this approach sacrifices the spatial resolution.

Another approach is to use a set of basis functions. Because we assume that the map is smooth and continuous, we should be able to sufficiently represent it using a relatively small number of bases. If J basis functions for each mode are enough to represent the map, then the map can be represented by the tensor $\underline{\mathbf{W}} \in \mathbb{R}^{J^M\times D}$, instead of $\underline{\mathbf{Y}} \in \mathbb{R}^{K^M\times D}$. Thus, $\underline{\mathbf{Y}}$ is *compressed* to $\underline{\mathbf{W}}$ by $(J/K)^M$. This approach also provides further advantages by using an orthonormal basis set. Because distances are preserved by the orthonormal transformation, the distance between two maps is equal to the distance between two corresponding coefficient vectors. Applying this to (50), the distance between $K^{M-1}\times D$ dimensional vectors can be evaluated by the distance

between $J^{M-1} \times D$ dimensional coefficient vectors. Accordingly, this reduces the calculation times in the competitive learning process. In this work, we used the normalized Legendre polynomials as a typical orthonormal system in the square latent space.

Another advantage of using a continuous basis set is that it provides the derivative of the objective function with respect to the latent variable $\mathbf{z}$. Thus we can find the best matching point using more efficient algorithms such as the gradient method. This means that we can specify the winners more precisely while using less computational time than in the ordinary all-play-all competition.

Suppose that the orthonormal system for mode m is $\left\{\varphi_1^{(m)}(\mathbf{z}), \ldots, \varphi_J^{(m)}(\mathbf{z})\right\}$, and let the matrix representation be $\mathbf{\Phi}^{(m)} = \left(\varphi_j(\boldsymbol{\zeta}_{k:})\right) \in \mathbb{R}^{K \times J}$. Then $\underline{\mathbf{Y}}$ is obtained by *decompressing* $\underline{\mathbf{W}}$ by

$$\underline{\mathbf{Y}} = \underline{\mathbf{W}} \times_1 \mathbf{\Phi}^{(1)} \times_2 \cdots \times_M \mathbf{\Phi}^{(M)} = \underline{\mathbf{W}} \times \{\mathbf{\Phi}\}. \tag{78}$$

In this paper, $\underline{\mathbf{W}}$ is called the core tensor of $\underline{\mathbf{Y}}$. To determine the winners, we also need the core tensors of the instance manifolds. Letting $\underline{\mathbf{V}}^{(m)}$ be the core tensor of $\underline{\mathbf{U}}^{(m)}$, the instance manifolds are obtained by

$$\underline{\mathbf{U}}^{(m)} = \underline{\mathbf{V}}^{(m)} \times_{-m} \{\mathbf{\Phi}\}. \tag{79}$$

The core tensors of the slice manifolds $\underline{\mathbf{T}}^{(m)}$ are easily calculated by only decompressing the $\underline{\mathbf{W}}$ with respect to mode m. Thus,

$$\underline{\mathbf{T}}^{(m)} \triangleq \underline{\mathbf{W}} \times_m \mathbf{\Phi}^{(m)}. \tag{80}$$

Because $\underline{\mathbf{T}}^{(m)}$ and $\underline{\mathbf{V}}^{(m)}$ are compressed representations for all modes except m, they are $(J/K)^{M-1}$ times smaller than the original $\underline{\mathbf{Y}}_{k(m)}$ and $\underline{\mathbf{U}}^{(m)}$. Nevertheless, the distance between the two maps is preserved by the orthonormal system. Thus,

$$\left\| \underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{U}}^{(m)}_{n(m)} \right\|^2 = \left\| \underline{\mathbf{T}}^{(m)}_{k(m)} - \underline{\mathbf{V}}^{(m)}_{n(m)} \right\|^2. \tag{81}$$

Applying (81) to (50), we can determine the winners without decompressing $\underline{\mathbf{W}}$ or $\underline{\mathbf{V}}$.

To obtain $\underline{\mathbf{W}}$ and $\underline{\mathbf{V}}$, we need to calculate the generalized inverse $\mathbf{\Phi}_G^{\#}$ for every iteration and mode. This computational cost could outweigh the benefits of using the basis functions. However, as described in Sec. 2, the inverse can be approximated by $\mathbf{\Phi}_P^{\#}$, which is just the transpose $\mathbf{\Phi}^{\mathrm{T}}$ in an orthonormal system. Thus, we do not need to calculate the inverse matrix at all.

Based on the above discussions, the TSOM algorithm with orthonormal bases is described below.

E step

Applying (81), the winner is determined without decompressing the core tensors. That is,

$$k_n^{*(m)} = \arg\min_k \left\| \underline{\mathbf{T}}^{(m)}_{k(m)} - \underline{\mathbf{V}}^{(m)}_{n(m)} \right\|^2. \tag{82}$$

We can replace the above all-play-all competition with a gradient method if preferred. In the case of the gradient descent method, it becomes

$$z_{nl}^{(m)} := z_{nl}^{(m)} - \eta \left\langle \underline{\mathbf{W}} \times_m \boldsymbol{\varphi}(\mathbf{z}_n^{(m)}) - \underline{\mathbf{V}}_{n(m)}^{(m)}, \underline{\mathbf{W}} \times_m \boldsymbol{\varphi}_l'(\mathbf{z}_n^{(m)}) \right\rangle. \tag{83}$$

After determining the winners, we calculate $\mathbf{B}^{(m)}$, $\mathbf{R}^{(m)}$, $\mathbf{G}^{(m)}$, and $\tilde{\mathbf{R}}^{(m)}$ using (51)–(54). Then, $\tilde{\mathbf{Q}}^{(m)}$ (the normalized responsibility of the bases) is calculated using

$$\tilde{\mathbf{Q}}^{(m)} = {\boldsymbol{\Phi}^{(m)}}^{\mathrm{T}} \tilde{\mathbf{R}}^{(m)}. \tag{84}$$

Here ${\boldsymbol{\Phi}^{(m)}}^{\mathrm{T}}$ is used as the generalized inverse.

M step

In the M step, the core tensors $\underline{\mathbf{W}}$ and $\{\underline{\mathbf{V}}^{(m)}\}$ are updated according to

$$\underline{\mathbf{W}} = \underline{\mathbf{X}} \times \left\{ \tilde{\mathbf{Q}} \right\} \tag{85}$$

$$\underline{\mathbf{V}}^{(m)} = \underline{\mathbf{X}} \times_{-m} \left\{ \tilde{\mathbf{Q}} \right\}. \tag{86}$$

The only difference between this and the discrete version in (55) (56) is that $\tilde{\mathbf{R}}^{(m)}$ is replaced by $\tilde{\mathbf{Q}}^{(m)}$. The binary tree calculation is also effective. The simulation results (Table 1, 2) show that the basis function is effective in reducing the calculation time.

3.6. Initialization and scheduling

TSOM is initialized in a similar way as conventional batch SOM. There are two random initialization methods: randomly initialize the reference vectors, or randomly initialize the winner. In the first case, the loop starts in the E step, whereas it starts in the M step in the second case. Although there are no significant differences, we recommend using the second method for TSOM, because it is not affected by biases in the given dataset. In real applications, initialization using PCA is also recommended. For relational data, PCA should be used for each mode, by vectorizing the slices of the data tensor. For example, a PCA for mode m regards $\hat{\mathbf{x}}_n^{(m)} = \mathrm{vec}\left(\underline{\mathbf{X}}_{n(m)}\right)$ as the nth data vector of mode m. When the PCA finishes, the initial winning nodes are determined using a principal axes transformation.

The time scheduling of the neighborhood size is implemented in a similar way. For example, $\sigma(t) = \max\left[\sigma_0(1 - t/\tau), \sigma_\infty\right]$ or $\sigma(t) = (\sigma_0 - \sigma_\infty)\, e^{-t/\tau} + \sigma_\infty$, where t is the calculation time, σ_0 and σ_∞ are the initial and last neighborhood sizes, and τ is the time constant.

3.7. TGTM

By applying the probabilistic generative model, it is possible to derive the algorithm using a fully Bayesian approach, namely, TGTM. In this paper, we have abbreviated TGTM for M-mode relational data to TGTM–R^M.

We first define the generative model for the relational data dealt with by TGTM–R^M. Suppose that we have M latent spaces $\{\mathfrak{Z}^{(m)}\}$ $(m = 1, \ldots, M)$, each of which is discretized to K_m nodes with the positional vectors $\{\boldsymbol{\zeta}_k^{(m)}\}$. The priors of the latent variables

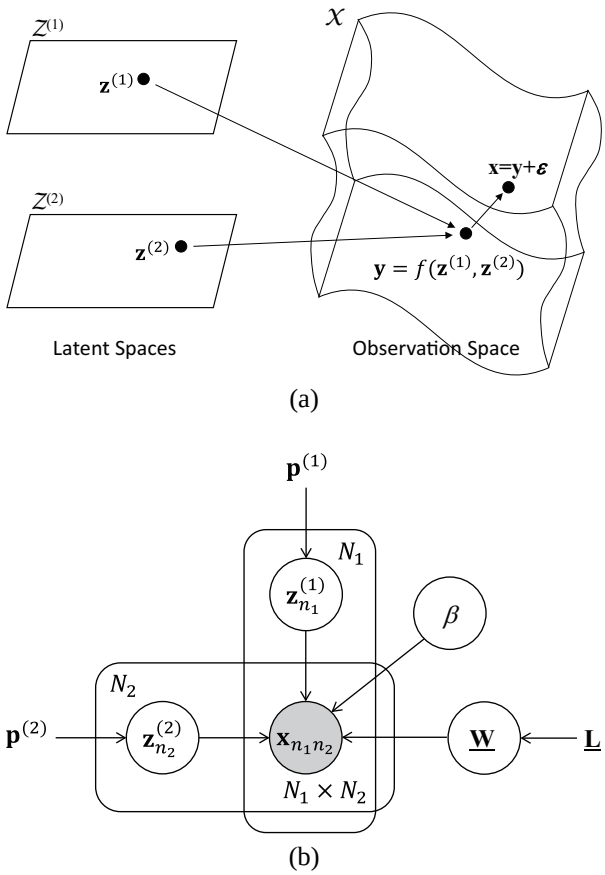


Figure 2: (a) The generative model of the relational data dealt with by TGTM–R^2. Note that the two 2-dimensional latent spaces are mapped onto the 4-dimensional manifold in the observation space, which is depicted as a 3-dimensional nonlinear cube. (b) The graphical model of the generative model.

are given by $\text{Prob}\left(\mathbf{z}^{(m)} = \boldsymbol{\zeta}_k^{(m)}\right) = p_k^{(m)}$, which we have assumed to be $p_k^{(m)} = 1/K_m$. To obtain the training dataset, the latent variables are generated N_m times, independently for each mode. The observations are made for all combinations of the latent variables using

$$\mathbf{x}_{n_1 \ldots n_M :} = \mathbf{y}\left(\mathbf{z}_{n_1:}^{(1)}, \ldots, \mathbf{z}_{n_M:}^{(M)}\right) + \boldsymbol{\varepsilon}_{n_1 \ldots n_M :}, \tag{87}$$

where $\mathbf{y}$ is the nonlinear map from the direct product space $\mathfrak{Z}^{(1)} \times \cdots \times \mathfrak{Z}^{(M)}$ to the observation space $\mathcal{X} \in \mathbb{R}^D$, and $\boldsymbol{\varepsilon} \in \mathbb{R}^D$ is the observation noise generated by $p(\boldsymbol{\varepsilon}) = \mathcal{N}\left(\boldsymbol{\varepsilon} \,\middle|\, \mathbf{0}, \beta^{-1}\mathbf{I}_D\right)$. Thus, $\prod_m N_m$ data are observed, and are represented by a tensor $\underline{\mathbf{X}} = (X_{n_1 \ldots n_M d})$.

We assume that the map is represented by a set of radial basis functions, such that

$$\underline{\mathbf{Y}} = \underline{\mathbf{W}} \times \{\boldsymbol{\Phi}\}, \tag{88}$$

where $\mathbf{\Phi}^{(m)} \in \mathbb{R}^{K_m \times J_m}$ is the radial bases of mode m, and $\underline{\mathbf{W}}$ is the core tensor. The prior of $\underline{\mathbf{W}}$ is

$$p\left(\underline{\mathbf{W}}\right) = \mathcal{N}\left(\underline{\mathbf{W}} \,\middle|\, \underline{\mathbf{O}}, {}^{-1}\underline{\mathbf{L}}\right) \tag{89}$$

$$\underline{\mathbf{L}} = (\lambda_1 \mathbf{I}_{K_1}) \circ \cdots \circ (\lambda_M \mathbf{I}_{K_M}). \tag{90}$$

Thus, each component of $\underline{\mathbf{W}}$ obeys the independent Gaussian prior $p(W_{j_1 \ldots j_M d}) = \mathcal{N}\left(W_{j_1 \ldots j_M d} \,\middle|\, 0, \lambda^{-1}\right)$. The generative model for $M = 2$ is shown in Fig. 2 (a) and (b).

The task of TGTM is to estimate the core tensor $\underline{\mathbf{W}}$, the latent variables $\{\mathbf{z}^{(m)}_{n_m:}\}$, and the parameter β from the observed relational data $\underline{\mathbf{X}}$. The TGTM algorithm can be derived by applying the standard EM algorithm. Here, $\underline{\mathbf{Y}}$ and β are estimated as the MAP solutions, and the latent variables $\{\mathbf{z}^{(m)}_{n_m:}\}$ are estimated as their posteriors. (Note that we can also easily estimate the posteriors of $\underline{\mathbf{Y}}$ and β by applying the variational Bayesian method [40].) Applying the variational approximation to this generative model, the objective function is

$$F(\underline{\mathbf{R}}, \underline{\mathbf{W}}, \beta) = -\frac{\beta}{2} E\left(\underline{\mathbf{X}}, \underline{\mathbf{Y}}(\underline{\mathbf{W}}); \{\mathbf{R}\}\right) - \sum_{\mathbf{k}} \sum_{\mathbf{n}} R_{\mathbf{kn}} \ln R_{\mathbf{kn}} + \ln P\left(\underline{\mathbf{W}}\right), \tag{91}$$

where $E(\underline{\mathbf{X}}, \underline{\mathbf{Y}}; \underline{\mathbf{R}})$ is the expectation error defined by (70), and $\mathbf{R}^{(m)}$ is the responsibility matrix of $\{\mathbf{z}^{(m)}_{n_m:}\}$. In the E step, F is maximized with respect to $\mathbf{R}^{(m)}$ using

$$\ln R^{(m)}_{kn} = -\frac{\beta}{2} E^{(-m)}_{kn}\left(\underline{\mathbf{X}}, \underline{\mathbf{Y}}(\underline{\mathbf{W}}); \{\mathbf{R}\}\right) + const. \tag{92}$$

If $M = 3$, (92) becomes

$$\ln R^{(1)}_{k_1 n_1} = -\frac{\beta}{2} \sum_{k_2, k_3} \sum_{n_2, n_3} R^{(2)}_{k_2 n_2} R^{(3)}_{K_3 n_3} \left\| \mathbf{y}_{k_1 k_2 k_3:} - \mathbf{x}_{n_1 n_2 n_3:} \right\|^2 + const. \tag{93}$$

In the M step, F is maximized with respect $\underline{\mathbf{W}}$ and then β using

$$\underline{\mathbf{W}} = \underline{\mathbf{X}} \times \left\{\tilde{\mathbf{Q}}\right\} \tag{94}$$

$$\beta^{-1} = \frac{1}{\prod_m N_m} E\left(\underline{\mathbf{X}}, \underline{\mathbf{Y}}(\underline{\mathbf{W}}); \{\mathbf{R}\}\right). \tag{95}$$

Here, $\tilde{\mathbf{Q}}^{(m)}$ is given by

$$\mathbf{G}^{(m)} = \mathrm{diag}\left(\sum_{n_m} R^{(m)}_{k_m n_m}\right) \tag{96}$$

$$\tilde{\mathbf{Q}}^{(m)} = \left(\mathbf{\Phi}^{(m)\mathrm{T}} \mathbf{G}^{(m)} \mathbf{\Phi}^{(m)} + \frac{\lambda_m}{\sqrt[M]{\beta}} \mathbf{I}_{K_M}\right)^{-1} \mathbf{\Phi}^{(m)\mathrm{T}} \mathbf{R}^{(m)} \tag{97}$$

. (98)

Table 1: Parameters used in the simulation of the artificial dataset, and the calculation results.

	Basic TSOM–R^2 (20 nodes/mode)	Legendre TSOM–R^2 with 4 bases/mode	TGTM–R^2 with 20 bases/mode	TGTM–R^2 with 4 bases/mode
Number of nodes (K_1, K_2)	20×20	—	20×20	20×20
Number of bases (J_1, J_2)	(20×20)	4×4	20×20	4×4
Neighborhood radius (σ_0)	2.0	2.0	2.0	2.0
(σ_∞)	0.1	0.1	0.8	0.8
Neighborhood time constant (τ)	50			
Number of instances (N_1, N_2)	100×100			
Noise amplitude (σ_{noise})	0.1			
RMSE (Average of 20 trials)	0.0775±0.0037	0.0867±0.0046	0.0693±0.0027	0.0777±0.0019
Calculation time for 100 loops (sec)*	0.45593±0.00083	0.09796±0.00027	15.3294±0.0057	10.9452±0.0038

* Intel Core i7-2600K (3.40GHz), Visual C++ (single thread)

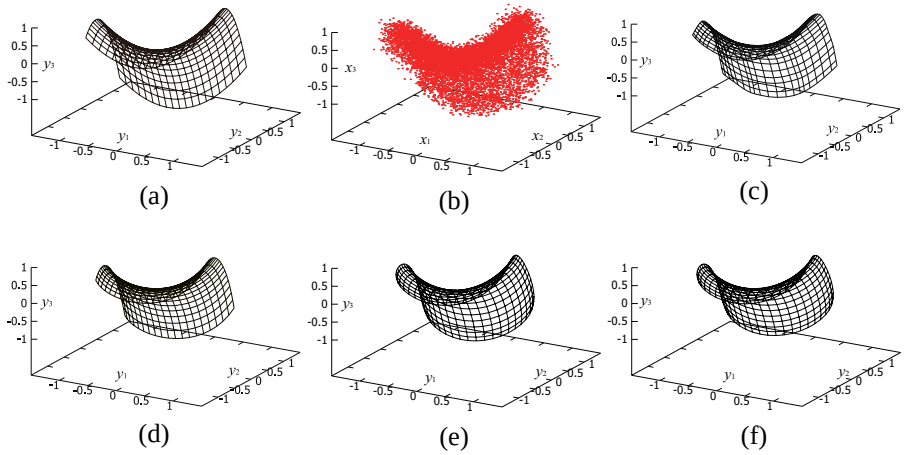


Figure 3: Results using the artificial dataset: (a) original map (desired result); (b) data points with Gaussian noise; (c) basic TSOM–R^2; (d) Legendre TSOM–R^2; (e) TGTM–R^2 with 20 RBFs; and (f) TGTM–R^2 with 4 RBFs.

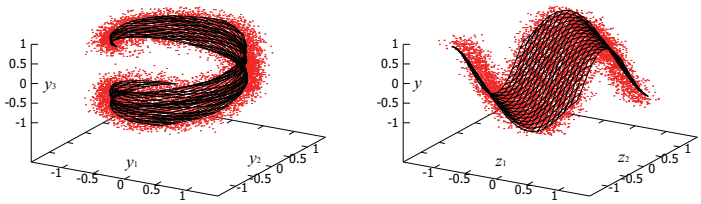
This is the TGTM–R^M algorithm. It is also possible to use the Gaussian process instead of the RBF bases. If the RBF bases are replaced by Nadaraya-Watson smoother, and the latent variables are estimated by the maximum likelihood, then this algorithm becomes the TSOM.

The TGTM algorithm is a Bayesian algorithm, which is its advantage compared with TSOM. However, the computational cost of TGTM is much greater than TSOM (Table 1, 2). Thus, it is hard to apply TGTM to practical tasks without the development of a more efficient algorithm. In this paper, the significance of TGTM is to bridge the gap between neural network-based algorithms and Bayesian algorithms, rather than to provide a practically useful algorithm. With such a solid theoretical background, we consider TSOM to be a widely applicable method that can be used with ease and confidence.

Table 2: Calculation time comparison for a large dataset.

	Basic TSOM–R^2	Legendre TSOM–R^2	TGTM–R^2
Number of nodes (K_1, K_2)	400×400	—	400×400
Number of bases (J_1, J_2)	(400×400)	16×16	16×16
Number of instances (N_1, N_2)	$1,000 \times 1,000$		
Calculation time for 100 loops (sec)*	2.65×10^3	3.38×10^1	3.91×10^5

* Intel Core i7-2600K (3.40GHz), Visual C++ (single thread)



(a) (b)

Figure 4: Results for the other artificial dataset organized by basic TSOM–R^2. Data points are indicated by dots. (a) 3-dimensional dataset and the organized map using TSOM–R^2. The axes represent the three components of the observation space. (b) 1-dimensional dataset and the organized map. Unlike other figures, the two horizontal axes represent the latent variables, and the observation space is only represented by the vertical axis.

4. Simulation Results

4.1. Artificial datasets

We used some artificial datasets to examine the performances of the proposed algorithms. For the sake of visualization, we synthesized 2-mode relational datasets with 1 dimensional latent spaces using the generative model. The protocol for generating data points is as follows. First, we generated N_m latent variables $\{z_1^{(m)}, \ldots, z_{N_1}^{(m)}\}$ using the uniform distribution in $[-1, +1]^L$ for $m = \{1, 2\}$. Then, the observation data $\mathbf{x}_{n_1 n_2}$ were generated by

$$\mathbf{x}_{n_1 n_2} = f\left(z_{n_1}^{(1)}, z_{n_2}^{(2)}\right) + \boldsymbol{\varepsilon}, \tag{99}$$

for $N_1 \times N_2$ combinations of $z_{n_1}^{(1)}$ and $z_{n_2}^{(2)}$. Here, $\boldsymbol{\varepsilon}$ is Gaussian noise $p(\boldsymbol{\varepsilon}) = \mathcal{N}\left(\boldsymbol{\varepsilon} \,\middle|\, \mathbf{0}, \sigma_{\text{noise}}^2 \mathbf{I}\right)$. In the simulations, we used nonlinear maps ranging in $[-1, +1]$, and set the noise amplitude to $\sigma_{\text{noise}} = 0.1$.

We examined four variations of TSOM and TGTM: (1) basic TSOM–R^2 with 20 discrete nodes per mode, (2) TSOM–R^2 with 4 Legendre bases per mode (Legendre TSOM–R^2), (3) TGTM–R^2 with 20 radial basis function (RBF) bases per mode, and (4) TGTM–R^2 with 4 RBF bases per mode. The parameters used in the simulation are shown in Table 1. TSOMs were initialized by randomly assigning the winners, and TGTM was initialized by randomly constructing the map. Empirically, GTM is more likely to fall into local minima than SOM, especially when it is initialized randomly. In this work, the radius of the RBF was gradually reduced to avoid local minima, in the same manner as the neighborhood size of the SOMs. With this modification, TGTM did not fall into local minima in any of the simulations. In the basic TSOM–R^2, the

latent variables are estimated by the all-play-all competition, whereas they are updated by the gradient descent method in the Legendre TSOM–R^2.

We iterated over the E and M steps, and reduced the neighborhood size exponentially with the time constant $\tau = 50$. In this experiment, the calculations were monitored until $t = 600$, but the map converged after a smaller number of iterations. Typically, at most one hundred iterations are enough for random initialization. The iteration cycles can be further reduced by using PCA initialization.

In the first artificial dataset, the data points were generated using

$$f(z_1, z_2) = \begin{pmatrix} z_1 \cos \pi/4 - z_2 \sin \pi/4 \\ z_1 \sin \pi/4 + z_2 \cos \pi/4 \\ z_1^2 - z_2^2 \end{pmatrix}. \tag{100}$$

The results are shown in Fig. 3 and Table 1. All the algorithms succeeded in capturing the data distribution, as expected. We repeated the simulations 20 times for each algorithm, randomly changing the initial state and the noise. All the results were consistent and stable. For this dataset, the RMSEs were more or less equal for all algorithms.

Among the examined algorithms, the fastest was Legendre TSOM–R^2. It was approximately 4 times faster than basic TSOM–R^2. In contrast, TGTM–R^2 was more than 50 times slower than the Legendre TSOM–R^2. The speed differences increased when the tasks became larger (Table 2). Therefore, Legendre TSOM–R^2 is the best solution for a large-scale relational data analysis.

Fig. 4 (a) shows the results using another artificial dataset, generated by

$$f(z_1, z_2) = \begin{pmatrix} \cos(0.5\pi z_1 + \pi z_2) \\ \sin(0.5\pi z_2 + \pi z_2) \\ z_2 \end{pmatrix}. \tag{101}$$

Ordinary SOM cannot generally learn this 'Swiss-roll' type of dataset. Nevertheless, TSOM succeeded. This is because the relational data contain more information about the latent variables than the ordinary dataset. This issue is discussed in Sec. 7.

TSOM can estimate the map even if the observed data are scalar. Fig. 4 (b) is an example of a scalar dataset. Here, the scalar data were generated by

$$f(z_1, z_2) = \sin\left(\frac{3\pi}{4}(z_1 + z_2)\right). \tag{102}$$

In this case, the two latent spaces degenerate into the one dimensional observation space. Thus, different latent variable pairs can often generate the same output. Nevertheless, TSOM estimated both the nonlinear map and the latent variables, as shown in Fig. 4 (b).

4.2. User–item rating datasets

As an example of a real application, we applied TSOM to two sets of user–item rating data: a sushi dataset with 2-modes, and a beverage dataset with 3-modes. The sushi dataset[2] consists of the ranking scores of 10 sushi toppings evaluated by 5,000

[2] http://www.kamishima.net/sushi/

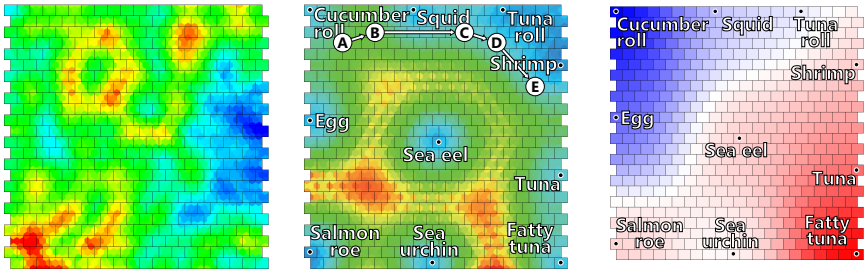


Figure 5: Maps for the sushi dataset: (a) the map of users (respondents) and (b) the map of items (sushi toppings), colored by the marginal U-matrix, where red regions represent the cluster borders and A–E are the conditioning points for Fig. 6; (c) the map of items (sushi toppings) colored by the marginal component plane. The red/blue colors indicate the higher/lower average scores. Fatty tuna is the most preferred topping, and cucumber roll is the least favorite.

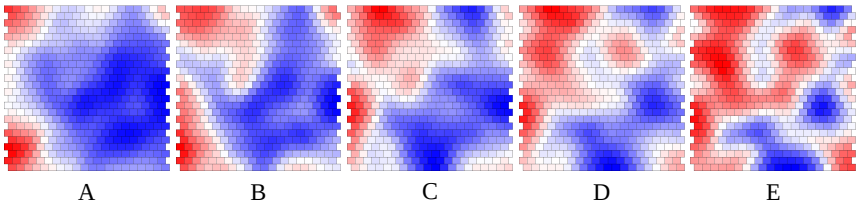


Figure 6: The user maps for the sushi dataset, colored by the conditional component plane. The conditioning points are labeled A–E in Fig. 5 (b). The red/blue regions in A correspond to the respondents who like/dislike cucumber rolls, while those in E show users who prefer the shrimp topping. When the conditioning point continuously moves from A to E in the sushi map, the user map changes color gradually from A to E.

Table 3: Summary of the sushi and beverage datasets.

	Sushi dataset	Beverage dataset
Number of modes	2 (user, item)	3 (user, item, context)
Number of instances	$5,000 \times 10$	$604 \times 14 \times 11$
Data type	scalar (integer)	scalar (integer)
Data scale	ranking from 1 to 10	grades from 1 to 5

respondents [41]. Thus, 10 toppings are ranked from 1 to 10 by each respondent. In this work, they are regarded as scalar scores. The beverage dataset[3] consists of scores evaluated by 604 respondents, who were asked to evaluate their preferences regarding 14 beverages (e.g., orange juice or coke) in 11 different contexts (e.g., at lunch or during sports). Thus, the beverage dataset is a 3-mode relational data of *(user)*×*(item)*×*(context)*. A summary of these datasets is shown in Table 3.

The results for the sushi dataset is shown in Fig. 5. Using TSOM–R^2, we obtained the topographic maps of users (respondents) and items (sushi toppings). These topographic maps can be translated as in conventional SOM. Therefore, two users or two items located closer to each other in the maps are more similar than ones that are fur-

[3]http://www.brain.kyutech.ac.jp/ furukawa/beverage-e/

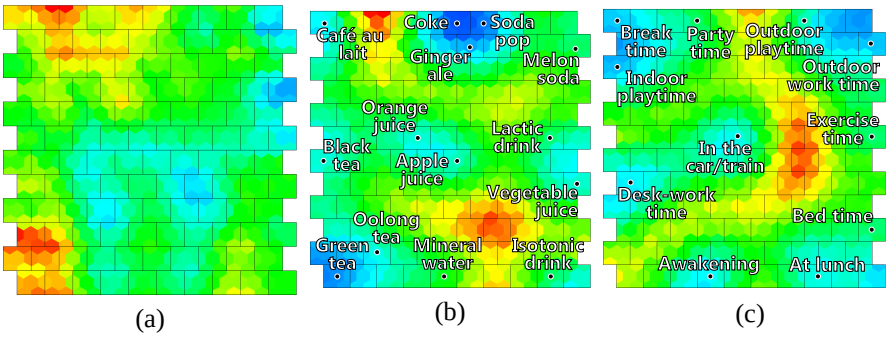


Figure 7: Maps for the beverage dataset, colored by the marginal U-matrix: (a) users (respondents); (b) items (beverages); and (c) contexts (situations).

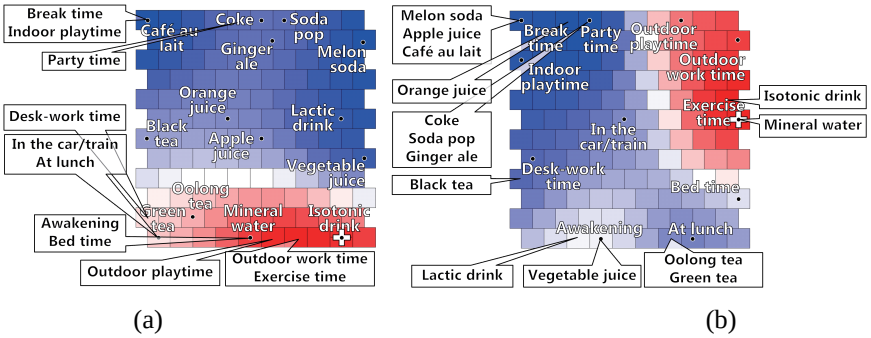


Figure 8: Maps of the beverage dataset colored by the conditional component plane. Red/blue regions represent the higher/lower scores. The markup balloons are the correspondence overlay. Note that the scores are marginalized with respect to the user mode. (a) The conditional component map of beverages colored for 'exercise time' ('+' in the context map). The beverages in the red region (mineral water and isotonic drink) are preferred during exercise. The overlaid label 'exercise time' is also located between these beverages. (b) The conditional component map of contexts colored for 'isotonic drink' ('+ in the beverage map). The map shows that isotonic drink is preferred in the situations indicated by red (exercise time, outdoor work time, and outdoor playtime). The overlaid label 'isotonic drink' is located near these situations.

ther away. For example, the shrimp sushi and tuna roll are closer in the map space. This implies that the people who prefer shrimp are also likely to prefer tuna rolls, and vice versa.

Fig. 7 presents the results for the beverage dataset obtained by TSOM–R^3. That is, the maps of users (respondents), items (beverages), and contexts (situations). Similar types of beverages are closer in the item map. For example, carbonated drinks are located in the same map region.

One may think that the difference between a TSOM and a conventional SOM is only due to the number of organized maps. However, it is worth noting that TSOM organizes a single model that preserves all the information, including the relationships between different modes. TSOM provides several visualization methods for the inter-mode relationships, as introduced in the next section.

5. Visualization and Exploration in Map Spaces

In this section, we introduce some visualization techniques for TSOM. We only describe the methods for TSOM, but they are also all available for TGTM.

5.1. The joint map

In conventional SOM, the organized nonlinear map is represented by a square topographic map. It is usually colored according to either the component plane or the U-matrix [42, 43]. The component plane visualizes the nonlinear map by focusing onto a component of the data vectors. Thus, nodes that have higher values with respect to the focused component are highlighted in the topographic map. Alternatively, when using the U-matrix, each node is colored depending on the distance in the data space from the node to its neighbors. As a result, the cluster boarders are highlighted in the topographic map. These two methods are also available for a TSOM.

For the TSOM, the purpose of visualizations is to outline the nonlinear map $\mathbf{y} = f(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)})$. The best way to capture the whole image is to visualize it for all combinations of the latent variables. Thus, we visualize f using a single topographic map, the dimension of which is ML. We refer to this as *the joint map*. Fig. 3 and Fig. 4 are examples of joint maps, which can also be represented by the component plane or the U-matrix. Unfortunately, limitations of the visualization mean that the joint map is only available when $M = 2$ and $L = 1$. Thus, we cannot visualize the whole image in most cases. Instead, we can color each topographic map to show the intra- and inter-mode relationships by focusing on an item of interest, as described in the following methods.

5.2. The marginal map

If our purpose is to analyze the relationships between instances within a mode, then *the marginal map* is appropriate. Using this method, we obtain M topographic maps, each of which is colored like a conventional SOM, taking the average over the unconcerned modes.

In the case of the marginal component plane, the kth node of the mth map is colored using a grayscale or heat map, which is dependent on the averaged value defined as

$$\overline{Y}_{kd}^{(m)} = \frac{1}{K_{-m}} \sum_{\substack{\mathbf{k} \\ k_m = k}} Y_{\mathbf{k}d}, \tag{103}$$

where d is the component of interest. Fig. 5 (c) shows the marginal component plane of the sushi map, which visualizes how much each topping is preferred by the respondents.

Similarly, for the marginal U-matrix, each node is colored using the average distance between two neighboring units, defined as

$$\overline{D}_m^2(k, k') = \frac{1}{K_{-m}} \left\| \underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{Y}}_{k'(m)} \right\|^2, \tag{104}$$

where k and k' are the indices of two neighboring nodes. Fig. 5 (a) (b) and Fig. 7 are examples using the marginal U-matrix.

5.3. The conditional map

The *conditional map* is useful if we want to analyze the relationships between two (or more) modes. Here, the term 'conditional' means that the topographic map of the focused mode is colored under the condition specified by the other modes.

When the nonlinear map $\mathbf{y} = f(\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ is visualized by the conditional map with respect to the 1st mode, f is regarded as a function of $\mathbf{z}^{(1)}$ under the conditions given by $\mathbf{z}^{(2)}$. Then, the topographic map of the 1st mode is colored by the component plane or U-matrix with respect to $\mathbf{z}^{(1)}$, while $\mathbf{z}^{(2)}$ is fixed to a specified position. Fig. 6 contains examples using the conditional component plane.

An interactive user interface on a PC can be used to take full advantage of the conditional map. We have developed prototype software, in which the user can specify the condition in one of the maps using a pointing device[4]. Then, the other maps are colored as conditional maps. When the user moves the mouse cursor in the conditioning map, then the other map changes accordingly, as if the user moves around in the space of the relational data (Fig. 6).

If $M > 2$, we can combine the marginal and conditional visualizations. Thus, we can color a focused map under the condition of some other modes, and then marginalize the remaining unconcerned modes. Fig. 8 (a) (b) contains some examples in which the component (rating score) is marginalized with respect to the user mode, while the other mode is used for conditioning.

5.4. Correspondence overlay

The interactive visualization allows for an intuitive analysis, but it is difficult to print. The *correspondence overlay* produces a printable static visualization, which summarizes the conditional component plane under different conditions. We can also use this technique to see the relationships between the instances belonging to two different modes.

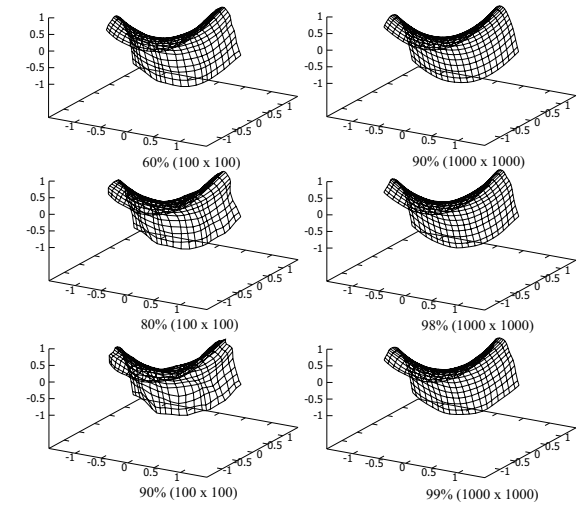
Fig. 8 (a) shows an example of the correspondence overlay, in which the context labels (markup balloons) are overlaid on the beverage map. These labels indicate the maximum point of the conditional component plane. For example, the label 'Exercise time' signifies the beverage with the highest score in this context. Thus, this method is similar to the correspondence analysis. Note that it is also possible to label the minimum points (i.e., the beverages with the worst scores in this context).

Conversely, Fig. 8 (b) is the correspondence overlay of beverage labels on the context map. In this case, the overlaid beverage labels mean the highest score point for the beverages in the context map. Simply speaking, Fig. 8 (a) shows the most preferred beverages for the labeled situations, whereas Fig. 8 (b) represents the most preferred situations for the labeled beverages. Thus, these two maps have different meanings.

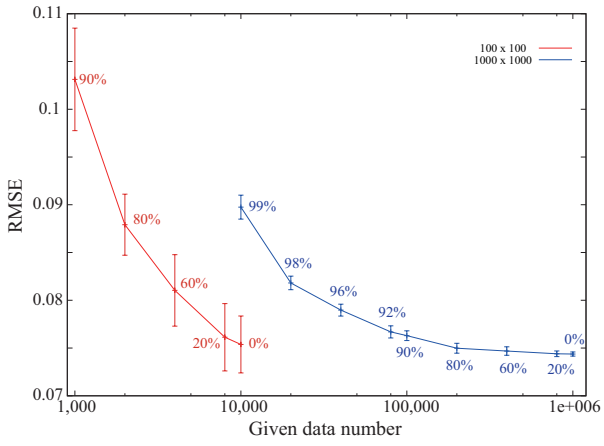
6. TSOM Variations

In this section, we introduce several variations of the TSOM, and example applications. We do not present the TGTM variations, but they are also possible.

[4] http://www.brain.kyutech.ac.jp/ furukawa/tsom-e/

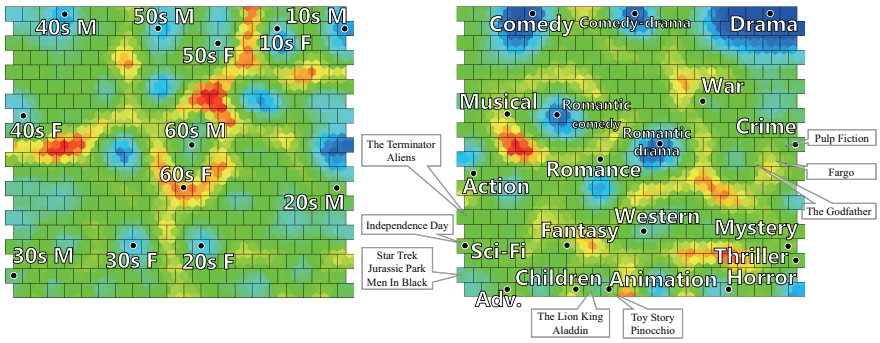


(a)



(b)

Figure 9: The results of the missing data estimation: (a) estimated manifolds under different missing ratios; and (b) RMSE of the missing data estimates. The RMSEs were evaluated for 2 different original data sizes, 100×100 and $1,000 \times 1,000$. Missing ratios are also indicated.



(a) (b)

Figure 10: The results for the MovieLens dataset: (a) user map (dominant classes of generations and genders are indicated by the correspondence overlay, 20s M and 20s F respectively represent men and women in their 20's); and (b) movie map (movie genres are indicated by the correspondence overlay, as well as some popular movie titles).

6.1. Missing data estimation

As previously discussed, user–item rating data is a typical application of TSOM. In real applications, customers usually purchase a small subset of the items, so most of the score table is missing. Missing value estimates are necessary when analyzing real data, and are also important when developing recommendation systems [5, 6].

Assume that there are two modes ($M = 2$), to simplify the explanation. When the given dataset contains missing values, the M step (47) becomes

$$Y_{k_1k_2d} = \frac{1}{G_{k_1k_2d}} \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} R^{(1)}_{k_1n_1} R^{(2)}_{k_2n_2} \Gamma_{n_1n_2d} X_{n_1n_2d}. \tag{105}$$

Here, $G_{k_1k_2d} = \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} R^{(1)}_{k_1n_1} R^{(2)}_{k_2n_2} \Gamma_{n_1n_2d}$, and $\Gamma_{n_1n_2d}$ indicates whether $X_{n_1n_2d}$ is missing. Thus, $\Gamma_{n_1n_2d} = 0$ when $X_{n_1n_2d}$ is missing, otherwise $\Gamma_{n_1n_2d} = 1$. The instance manifolds are also calculated in a similar way.

In the E step, we can determine winners in two ways. One way is to use (77), as in the ordinary case. This method results in a quick winning decision. When the missing ratio increases, there may only be a few data points for each instance. In such cases, it is better to directly evaluate (76), because the distance between the instance and the slice manifolds may be affected by missing data. After the training process, the missing values are estimated using $\tilde{X}_{n_1n_2d} = Y_{k^{*(1)}_{n_1} k^{*(2)}_{n_2} d}$.

We investigated this algorithm using the same artificial dataset as in Fig. 3. We randomly removed part of the dataset, and then passed it to the TSOM. (At least 2 data points were left for each instance). In this simulation, we estimated the winning positions using (76). We repeated this protocol, changing the various missing ratios. The results are shown in Fig. 9. The TSOM captured the manifold shape, even when the missing ratio was higher than 90% (Fig. 9 (a)). The RMSE result also shows that TSOM was robust to missing data (Fig. 9 (b)).

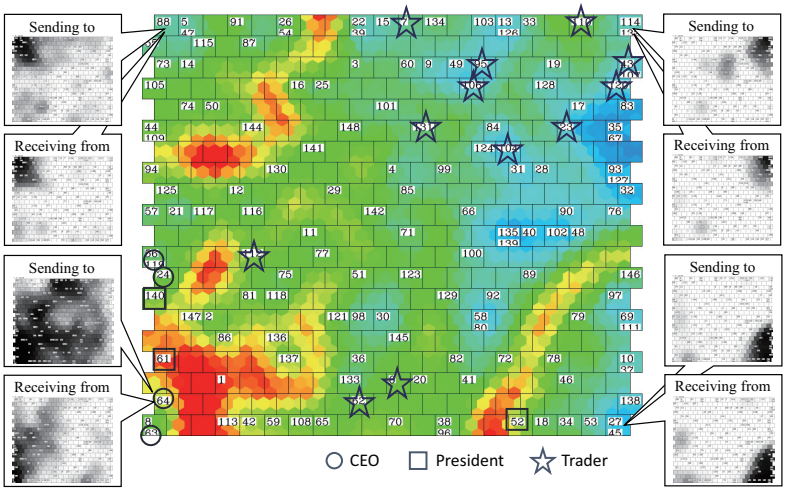


Figure 11: The results for the Enron e-mail dataset. The center map is colored by the marginal U-matrix, where the cluster borders are indicated by red regions. The numbers represent user IDs. The surrounding small maps are the conditional component planes, representing the sending/receiving e-mail traffic from/by the conditioned user.

We also applied this algorithm to the MovieLens dataset[5], which contains user–item rating data for movie titles. The dataset used here is a subset called MovieLens 100k, which contains 100,000 rating scores on a 5-point scale. The scores were evaluated by 943 people for 1,682 movie titles, and approximately 94% of the data are missing. The dataset also contains the information for the respondent attributes and the movie title genres, but they were not used in this experiment. To evaluate the missing data estimation, we used the training and test dataset provided by MLcomp[6]. The root mean square error (RMSE) was 0.9533 ± 0.0008. This result more or less equals to other methods within top 10 in MLcomp.

Here, we introduced a simple algorithm for estimating missing data. By applying the EM algorithm, this algorithm may be integrated into the TGTM, so that the missing values can be estimated at the same time as the nonlinear map. Such a probabilistic approach is an important issue for further investigation.

6.2. Using side information

In some situations, some additional data called side information are also provided. For example, the MovieLens dataset contains the side information on the age, gender, and occupation as user attributes. In such cases, it is possible to consider side information with the relational data.

Suppose that $\mathbf{x}_n^{\text{s.i.}(m)}$ is the side information of mode m. Then, the winner of mode m is determined using both the instance manifold and the side information. Let $\mathbf{y}_k^{\text{s.i.}(m)}$

[5]http://grouplens.org/datasets/movielens/
[6]http://www.mlcomp.org/

be the kth reference vector of the side information of mode m, and let $D^{\text{s.i.}(m)}$ be the dimension of $\mathbf{x}_n^{\text{s.i.}(m)}$. Then, the winner is determined by

$$k_n^{*(m)} = \arg\min_k \left\{ \frac{1}{K_{-m}D} \left\| \underline{\mathbf{Y}}_{k(m)} - \underline{\mathbf{U}}_{n(m)}^{(m)} \right\|^2 + \frac{\alpha^{(m)}}{D^{\text{s.i.}(m)}} \left\| \mathbf{y}_k^{\text{s.i.}(m)} - \mathbf{x}_n^{\text{s.i.}(m)} \right\|^2 \right\}. \tag{106}$$

Here, $\alpha^{(m)}$ is a weight parameter that determines how much the side information affects the winning decision. A typical value for $\alpha^{(m)}$ is 1, so the relational data and the side information equally effect the winning decision. In the M step, $\mathbf{y}_k^{\text{s.i.}(m)}$ is also updated according to

$$\mathbf{Y}^{\text{s.i.}(m)} = \tilde{\mathbf{R}}^{(m)} \mathbf{X}^{\text{s.i.}(m)}. \tag{107}$$

Thus, the TSOM works as an ordinary SOM with respect to the side information.

We applied this algorithm to the MovieLens dataset. The side information for the respondents (age, gender, and occupation) and the movie titles (genres such as drama, comedy, SF, etc.) were considered in the winning decision for $\alpha^{(m)} = 1$. The obtained maps are shown in Fig. 10. The user map showed distinct cluster borders, which coincided with users' generations and genders. Some clusters of genres were also present in the movie map.

6.3. TSOM for square relational data

For M-mode relational data, each mode corresponds to an object set, which is the item of interest for our analysis. Thus, there are typically M object sets to be analyzed. However, some modes may share the same object set. For example, there is only one object set to be analyzed, and each observed data point is determined by a combination of two members of the set. In this case, the observed data are modeled by a nonlinear function, $\mathbf{x}_{n_1 n_2 :} \simeq f(\mathbf{z}_{n_1}, \mathbf{z}_{n_2})$, where both $\mathbf{z}_{n_1}, \mathbf{z}_{n_2}$ belong to the same latent space, $\mathfrak{Z}$. We refer to this type of relational data as square, because the data tensor becomes a square with respect to the two modes. This situation can be generalized to cases in which three or more modes share an identical object set.

A representative example application is message traffic data between members of the same community (e.g., e-mail traffic within a company). In this case, the sender set is identical to the receiver set. When applied to this problem, basic TSOM–R^2 organizes two different maps (i.e., the sender and receiver maps), which contain different arrangements of the community members. However, if we wish to analyze mutual interactions between members, we need a single topographic map representing the member relationships within the community. To obtain such a unified map, the winner should be determined with respect to both modes. Thus, the winner of each member is defined by

$$k_n^* = \arg\min_k \left(\left\| \mathbf{Y}_{k::} - \mathbf{U}_{n::}^{(1)} \right\|^2 + \left\| \mathbf{Y}_{:k:} - \mathbf{U}_{:n:}^{(2)} \right\|^2 \right), \tag{108}$$

instead of (28) – (30).

We applied this algorithm to the e-mail data of the Enron dataset[7] [44, 45]. This dataset consists of e-mails from 148 Enron employees. The procedure for the experiment is summarized as follows. The relational data used here is the number of e-mails between members, determined using the 'To:' field. Because the distribution has a long tail, we transformed the traffic number using the exponential function so that the data were approximately uniformly distributed in $[0, 1]$. e-mail traffic to oneself was ignored, and treated as a missing value.

The results are shown in Fig. 11. We can recognize some clusters using the marginal U-matrix (center map). The most distinct cluster represents the executives, whose e-mail patterns are quite different from the other members. The traffic between two nodes of the map can be visualized using the conditional component planes (small maps). These results show that e-mails were mainly exchanged within each cluster, suggesting that the clusters correspond to company departments.

In this section, we presented two practical applications to demonstrate the potential of the TSOM. Needless to say, further investigations are necessary to examine the validity of the obtained maps. What we wish to emphasize here is that TSOM can be extensively applied to many fields with some modifications, and can extract rich information by combining visualization techniques.

7. Discussion: The Tensor SOM Family

In this section, we compare TSOM with another SOM variation that is relevant to a tensorial representation. This discussion is also an attempt at constructing a theoretical background for the TSOM family. Furthermore, the significance of TSOM as a generalization of the conventional SOM is also discussed.

7.1. Comparison with SOM of SOMs (SOM^2)

Among SOM variations, SOM^2 (or 'SOM of SOMs') is the most related to our method [33, 34]. TSOM and SOM^2 are very alike, and there are many similarities in their tasks and algorithms. First, the common aim of both SOMs is to estimate the nonlinear map from two or more latent spaces to the observation space, which is represented by a tensor. Second, the winner for each instance is determined by the distance from the instance manifold to the slice manifolds. (In the original paper, these are called *'sections'*, and the other orthogonal manifolds are *'fibers'* because of the fiber bundle term [34]). However, the most essential difference relates to their hierarchical structures. There is a distinct hierarchy between modes which are referred to as 'parent' and 'children' in SOM^2, whereas there is no such hierarchy in TSOM. The difference originates in the type of data structures used by these algorithms. Thus, these two algorithms share the same goal (estimating the nonlinear map from the multiple latent spaces to the observation space), but use different start points (the data structure of the given dataset). In this sense, it is better to call SOM^2 a *'tensor SOM for hierarchical data'* (TSOM-H), and call the proposed algorithm a *'tensor SOM for relational data'*

[7] https://www.cs.cmu.edu/~./enron/

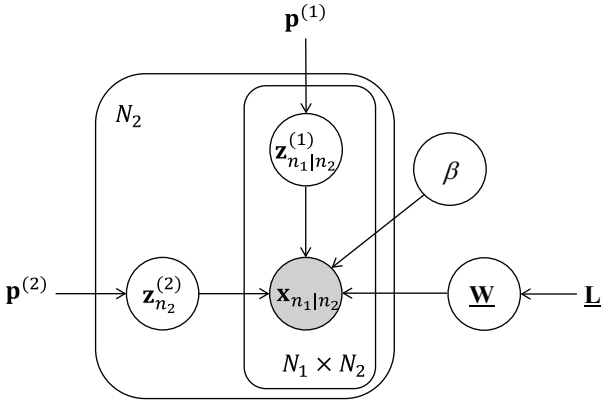


Figure 12: Graphical representation of the generative model for SOM^2 (TSOM–H^2).

(TSOM-R). Clarifying the similarities and differences between the two algorithms is important when comprehensively constructing the family of TSOMs. Hereafter, SOM^2 is referred as TSOM-H and is regarded as a member of the TSOM family.

7.2. TSOM–H^2 (SOM^2) algorithm

The essential difference resides in the data structure, so let us start from the generative model of SOM^2, i.e., TSOM–H^2. Suppose that we have two object sets, $\Omega^{(1)}$ and $\Omega^{(2)}$, and suppose that $\Omega^{(2)}$ is the parent of $\Omega^{(1)}$ in the data observation. Here 'parent' means that it behaves like a parameter rather than a variable, and a series of observations is made for various child objects under the same parent object. For example, if we independently conduct consumer surveys for every new product, then the product set becomes the parent of the respondent set. Note that the obtained dataset is no longer relational, because different respondents are sampled for every survey. Therefore, we do not know how the same respondent evaluates other products.

Under this situation, the observed dataset is obtained in the following way. First, we sample a set of instances of the parent mode, $\Omega^{(2)}_S = \{\omega^{(2)}_{n_2}\}$, and then independently sample the instances of the child mode for each $\omega^{(2)}_{n_2}$. Thus, we have N_2 sample sets with respect to mode 1, $\Omega^{(1)}_{S|\omega^{(2)}_{n_2}} = \left\{\omega^{(1)}_{n_1|n_2}\right\}$. Note that $\omega^{(1)}_{n_1|n_2} \neq \omega^{(2)}_{n_1|n'_2}$ if $n_2 \neq n'_2$. As a result, we obtain $N_1 \times N_2$ data $\{\mathbf{x}_{n_1|n_2}\}$ from the combination of $\left(\omega^{(1)}_{n_1|n_2}, \omega^{(2)}_{n_2}\right)$. This generative model is shown in Fig. 12. (Strictly speaking, the sample number N_1 is also determined independently for each $\omega^{(2)}_{n_2}$, but we assume that they are equal for the ease of explanation).

Considering the above discussion, the SOM^2 algorithm can be rewritten as TSOM–H^2, a variation of TSOM. Although the algorithm described below looks quite different from the original, they are equivalent. In the E step, the winner is determined according

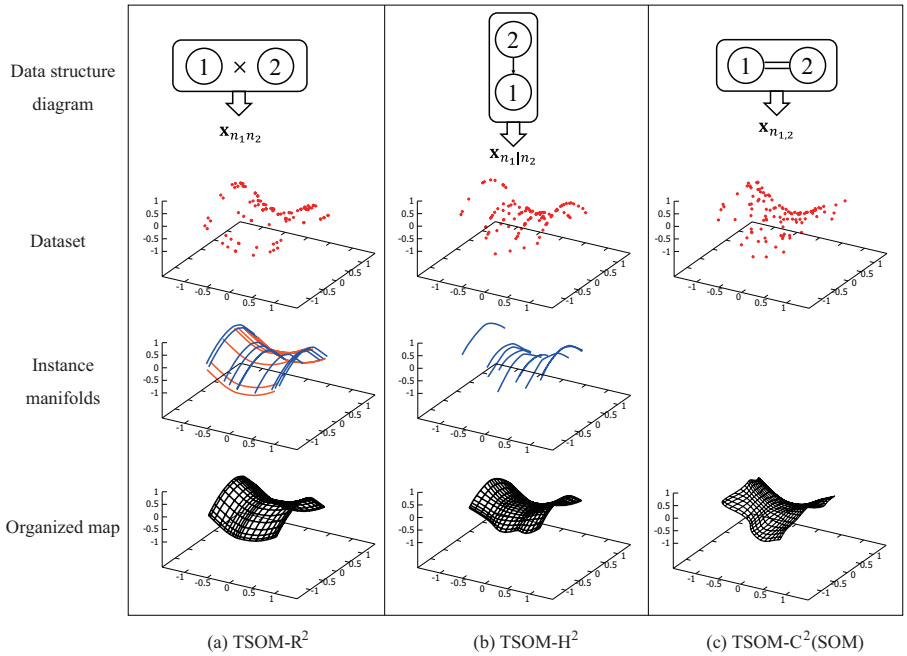


Figure 13: Comparison of three possible data structures and their corresponding algorithms: (a) relational data with TSOM–R^2; (b) hierarchical data with TSOM–H^2; and (c) coupled data with TSOM–C^2 (which becomes ordinary SOM). The 1st row contains the data structure diagrams, and the 2nd shows examples of the sampled dataset. The relational data in (a) are aligned in two directions, while the hierarchical data in (b) are aligned in one direction. No data alignment was applied to the coupled data case in (c). The 3rd row contains the instance manifolds calculated by TSOM–R^2 and TSOM–H^2. No instance manifold is estimated in TSOM–C^2. The 4th row contains the estimated nonlinear maps.

to

$$k_{n_1|n_2}^{*(1)} = \arg\min_{k_1} \left\| \mathbf{y}_{k_1 k_{n_2}^{*(1)}:} - \mathbf{x}_{n_1|n_2} \right\|^2 \tag{109}$$

$$k_{n_2}^{*(2)} = \arg\min_{k_2} \left\| \mathbf{Y}_{:k_2:} - \mathbf{U}_{:n_2:}^{(2)} \right\|^2 . \tag{110}$$

(109) and (110) mean that each winner of the parent mode is determined by the distance between $\mathbf{U}_{:n_2:}^{(1)}$ (the instance manifold) and $\mathbf{Y}_{:k_2:}^{(2)}$ (the slice manifolds), whereas the winner of the child mode is determined by the best matching unit within the best matching slice manifold of its parent.

After determining the winners, the responsibility matrices $\mathbf{R}_{n_2}^{(1)}$ and $\mathbf{R}^{(2)}$ are calculated using the neighborhood matrices $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$, and the winning matrices $\mathbf{B}_{n_2}^{(1)}$ and $\mathbf{B}^{(2)}$. That is,

$$\mathbf{R}_{n_2}^{(1)} = \mathbf{H}^{(1)} \mathbf{B}_{n_2}^{(1)} \tag{111}$$

$$\mathbf{R}^{(2)} = \mathbf{H}^{(2)} \mathbf{B}^{(2)}. \tag{112}$$

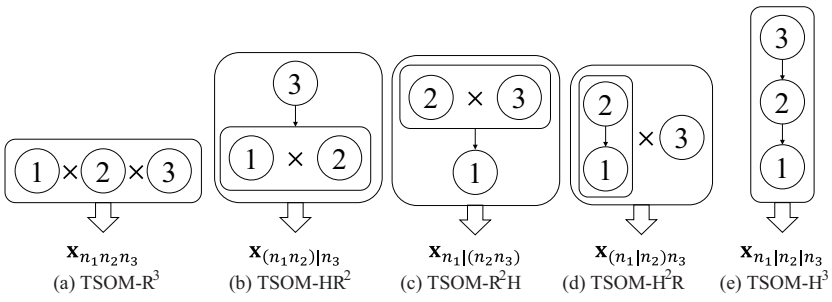


Figure 14: Diagrams of the possible 3rd order data structures.

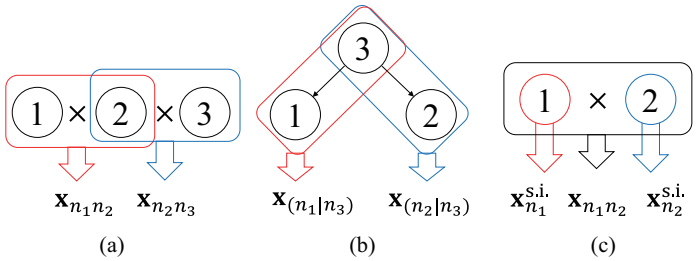


Figure 15: Three examples of data structures with two or more observation spaces. Note that there are more possible data structures. The data structure in (c) corresponds to relational data with side information.

Note that the responsibility of the child mode $\mathbf{R}^{(1)}_{n_2}$ depends on its parent $\omega^{(2)}_{n_2}$, whereas the responsibility of the parent $\mathbf{R}^{(2)}$ does not depend on its child mode.

In the M step, the map is updated according to

$$\mathbf{U}^{(2)}_{:n_2:} = \mathbf{X}_{:n_2:} \times_1 \tilde{\mathbf{R}}^{(1)}_{n_2} \tag{113}$$

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}^{(2)} \times_2 \tilde{\mathbf{R}}^{(2)}. \tag{114}$$

We iterate over these two steps until the map converges.

Fig. 13 shows how TSOM–R^2 and TSOM–H^2 estimate the nonlinear map from the given datasets. In TSOM–R^2, the instance manifolds are estimated for every mode, whereas in TSOM–H^2, they are only estimated for the parent mode.

7.3. Data structure diagram

To illustrate the data structures, let us introduce the diagram in Fig. 13 (top row). In this diagram, each sample set is represented by a circle node and its mode number. When two or more nodes are integrated into a set of sample combinations, they are packaged to an oval node. The nodes with no links mean that the samples of the nodes were selected independently to other nodes. The symbol '×' represents the direct product of independent nodes, which produces all possible combinations of two sample sets. The black thin arrow denotes the hierarchical sampling of the connected nodes, and the nodes connected by double lines are coupled in the data sampling. Thus, a

pair of instances is always sampled together. The thick white arrow represents the data observation, which is made for every member of the root node.

The top row of Fig. 13 (a) contains a diagram of a 2-mode relational dataset, dealt with by TSOM–R^2. In this case, the data vectors $\{\mathbf{x}_{n_1 n_2}\}$ are observed for all members of the sample set $\Omega_S = \left\{\left(\omega_{n_1}^{(1)}, \omega_{n_2}^{(2)}\right)\right\}$.

Fig. 13 (b) depicts the 2-mode hierarchical dataset case, dealt with by TSOM–H^2. In this case, the data vectors $\{\mathbf{x}_{n_1|n_2}\}$ are observed for all members of $\Omega_S = \left\{\left(\omega_{n_1|n_2}^{(1)}, \omega_{n_2}^{(2)}\right)\right\}$. This situation appears in transfer or multisystem learning tasks, where the parent mode corresponds to the system parameter, and the child mode represents the state variable under the given parameter [46, 47].

In the last situation depicted in Fig. 13 (c), two nodes are tightly coupled in the data observation. In this case, a pair of instances $\left(\omega_n^{(1)}, \omega_n^{(2)}\right)$ is sampled simultaneously, and then a data vector $\mathbf{x}_n$ is observed from the instance pair. Here, we refer to the TSOM for this type dataset as TSOM–C^2 (tensor SOM for coupled data). Interestingly, the TSOM–C^2 algorithm is almost the same as the conventional SOM, because we do not choose the winning position but take the ordinary best matching point in the entire nonlinear map. This situation corresponds to nonlinear ICA, and several previous studies have shown that SOM can be applied to this task [48, 49]. It would be worth investigating TSOM–C from the viewpoint of nonlinear ICA. This is an important issue that we will consider in the future.

The difficulties of these tasks are also different for the three cases. In case (a), TSOM–R^2 must estimate $(N_1 + N_2)L$ latent variables from $N_1 N_2 D$ observed data, whereas TSOM–H^2 in case (b) must estimate $(N_1 N_2 + N_2)L$ unknowns from $N_1 N_2 D$ known values. Thus, TSOM–H^2 task is more difficult than TSOM–R^2. The most difficult is case (c), in which TSOM–C^2 must estimate $2NL$ latent variables from ND observed data. This is why TSOM–R^2 captured the data distribution better than ordinary SOM (i.e., TSOM–C^2).

7.4. TSOM family of order 3

If there are 3 or more modes, new situations arise in which the relational and the hierarchical data structures are mixed. For example, we first obtain the sample set of mode 1, and then observe the relational data of modes 2 and 3 for each sample of mode 1. In this case, the data vectors are denoted by $\mathbf{x}_{(n_2 n_3)|n_1}$. For 3-mode data, there are 5 possible data structures, as shown in Fig. 14. Here, we have excluded the data structures with coupled modes. The corresponding TSOMs are (a) TSOM–R^3, (b) TSOM–HR^2, (c) TSOM–R^{2}H, (d) TSOM–H^{2}R, and (e) TSOM–H^3 (SOM^3). All these algorithms can be derived by combining TSOM–R and TSOM–H. For example, the TSOM–HR^2 algorithm (Fig. 14 (b)) can be described as follows.

E step

$$k^{*(1)}_{n_1|n_3} = \arg\min_{k_1} \left\| \underline{\mathbf{Y}}_{k_1:k^{*(3)}_{n_3}:} - \underline{\mathbf{U}}^{(1)}_{n_1::|n_3} \right\|^2 \tag{115}$$

$$k^{*(2)}_{n_2|n_3} = \arg\min_{k_2} \left\| \underline{\mathbf{Y}}_{:k_2k^{*(3)}_{n_3}:} - \underline{\mathbf{U}}^{(2)}_{:n_2:|n_3} \right\|^2 \tag{116}$$

$$k^{*(3)}_{n_3} = \arg\min_{k_3} \left\| \underline{\mathbf{Y}}_{::k_3:} - \underline{\mathbf{U}}^{(3)}_{::n_3:} \right\|^2 \tag{117}$$

M step

$$\underline{\mathbf{U}}^{(1)}_{n_1::|n_3} = \underline{\mathbf{X}}_{::n_3:} \times_2 \tilde{\mathbf{R}}^{(2)}_{n_3} \tag{118}$$

$$\underline{\mathbf{U}}^{(2)}_{:n_2:|n_3} = \underline{\mathbf{X}}_{::n_3:} \times_1 \tilde{\mathbf{R}}^{(1)}_{n_3} \tag{119}$$

$$\underline{\mathbf{U}}^{(3)}_{n_3} = \underline{\mathbf{X}}_{::n_3:} \times_1 \tilde{\mathbf{R}}^{(1)}_{n_3} \times_2 \tilde{\mathbf{R}}^{(2)}_{n_3} \tag{120}$$

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}^{(3)} \times_3 \tilde{\mathbf{R}}^{(3)} \tag{121}$$

Thus, two types of E and M steps are combined with respect to the data structure. The algorithms for the other cases can be easily obtained in a similar way.

In some cases, there are two or more observation spaces. Some examples are presented in Fig. 15. In this figure, the data structure (a) is the case in which two different surveys are made for the same group of respondents. The data structure of the MoveiLens with side information is represented by (c). When the number of modes increases, the number of possible variations increases exponentially. Nevertheless, the TSOM (and the TGTM) family can adapt to all data structures. Therefore, these algorithms can be unified using a comprehensive theoretical system of topographic mapping families for tensorial data.

7.5. The relationship between the original SOM and TSOM

Finally, let us discuss the original SOM from the viewpoint of TSOM. Because the original method can be regarded as a special case of TSOM, it is expected that TSOM provides several new perspectives on the original SOM.

In the original SOM, the winning node is determined to be the node in the data space nearest to a given data vector. Therefore the learning result depends on the metric that determines distances in the data space. It is usual for every component of the data vectors to be normalized so that their mean and variance are zero and one respectively, and the Euclidean distance is applied. This protocol seems to treat all components equally, but the metric problem still remains. If some components are almost identical to each other, then those duplicated components may affect the metric more than others. For example, if a teacher tries to analyze students' abilities using data consisting of many physical indices and few academic indices, then academic ability is almost ignored as noise. This metric problem is generally unavoidable in unsupervised learning methods. Even though TSOM is also affected by this problem, TSOM can

reduce these undesired effects. By regarding the dataset as a 2-mode relational dataset, TSOM generates two maps, namely a map of target objects and a map of data components. If two components are identical, then those components are located in the same position in the component map. Consequently, their influence in determining winners is weakened, and thus TSOM generates a more moderate result. Therefore, TSOM is recommended for use in conventional tasks where the original SOM has been applied.

The second perspective is relevant to the origins of the SOM. The SOM was originally a neural network model of a brain map of a visual field, and turned out to be useful as a dimension reduction tool for practical applications [2]. Early studies investigated both the self-organization of topology preserving neural projections in visual fields (e.g., [50]), and the self-organization of visual features (e.g., [51]). Though those studies appear contiguous, there is a gap between them. The former is the self-organization of the order of visual signal lines, whereas the latter orders the features of visual signals. Kohonen referred to these as type-I and type-II self-organization respectively [52]. From the viewpoint of the TSOM, type-I corresponds to a map of components, while type-II is a map of objects. In this sense, TSOM unifies both types of self-organization.

The third perspective is relevant to the axes of the topographic map. Usually the meaning of the axes are of little concern. However, by regarding the conventional 2-dimensional square map as a product space of two 1-dimensional latent spaces, the SOM becomes a TSOM for coupled data, that is, TSOM–C^2. As has already been discussed above, this is related to nonlinear ICA. This is an important issue for TSOMs that should be investigated further.

8. Conclusions

In this paper, we presented two algorithms for topographic mappings of tensorial data, TSOM and TGTM. TGTM provides the theoretical background of the algorithms, and TSOM is useful for practical applications. Among the variations, the TSOM with an orthonormal bases was computationally fast and had a high resolution. Therefore, it is the best choice for large-scale tasks. We also presented various TSOM variations and visualization methods, which further increase the applicability of TSOM to real data analysis.

Theoretically, TSOM and TGTM can be derived from the generative model by applying the EM algorithm. SOM^2 (SOM of SOMs) is a sibling of TSOM that is adapted to a hierarchical data structure. We have shown that SOM^2 can be unified to the TSOM family. Therefore, this work presented some nonlinear tensor analysis methods, and attempted to establish a comprehensive theoretical system for the family of algorithms.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers 23500280, 22120510.

[1] T. Kohonen, Self-Organizing Maps, Springer-Verlag, Berlin Heidelberg, 2001.

[2] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1) (1982) 59–69.

[3] C. M. Bishop, M. Svensen, C. K. I. Williams, GTM: The generative topographic mapping, Neural Computation 10 (1998) 215–234.

[4] C. M. Bishop, M. Svensen, C. K. I. Williams, Developments of the generative topographic mapping., Neurocomputing 21 (1-3) (1998) 203–224.

[5] G. Ricci, M. de Gemmis, G. Semeraro, Matrix and tensor factorization techniques applied to recommender systems: a survey, International Journal of Computer and Information Technology 1 (2012) 94–98.

[6] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37. doi:10.1109/MC.2009.263.

[7] A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, ACM, New York, NY, USA, 2010, pp. 79–86. doi:10.1145/1864708.1864727.

[8] E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy, B. Yener, Modeling and multiway analysis of chatroom tensors, in: Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics, ISI'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 256–268.

[9] E. Acar, S. A. Camtepe, B. Yener, Collective sampling and analysis of high order tensors for chatroom communications, in: in ISI 2006: IEEE International Conference on Intelligence and Security Informatics, Springer, 2006, pp. 213–224.

[10] M. W. Berry, M. Browne, Email surveillance using non-negative matrix factorization., Computational & Mathematical Organization Theory 11 (3) (2005) 249–264.

[11] B. W. Bader, M. W. Berry, M. Browne, Discussion tracking in Enron email using PARAFAC, in: M. W. Berry, M. Castellanos (Eds.), Survey of Text Mining II: Clustering, Classification, and Retrieval, Springer, 2008, Ch. 8, pp. 147–163.

[12] F. Miwakeichi, E. Martínez-Montes, P. A. Valdeś-Sosa, N. Nishiyama, H. Mizuhara, Y. Yamaguchi, Decomposing EEG data into space-time-frequency components using parallel factor analysis, NeuroImage 22 (3) (2004) 1035 – 1045.

[13] J. Li, L. Zhang, D. Tao, H. Sun, Q. Zhao, A prior neurophysiologic knowledge free Tensor-Based scheme for single trial EEG classification, Neural Systems and Rehabilitation Engineering, IEEE Transactions on 17 (2) (2009) 107–115.

[14] M. A. O. Vasilescu, D. Terzopoulos, Multilinear image analysis for facial recognition, in: ICPR (2), 2002, pp. 511–514. doi:10.1109/ICPR.2002.1048350.

[15] J. Yang, D. Zhang, A. F. Frangi, J.-y. Yang, Two-dimensional PCA: A new approach to appearance-based face representation and recognition, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 131–137. doi:10.1109/TPAMI.2004.1261097.

[16] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, H. Zhang, Multilinear discriminant analysis for face recognition, IEEE Trans. on Image Processing 16 (1).

[17] N. E. Helwig, S. Hong, J. D. Polk, Parallel factor analysis of gait waveform data: A multimode extension of principal component analysis, Human Movement Science 31 (3) (2012) 630 – 648.

[18] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM REVIEW 51 (3) (2009) 455–500.

[19] E. Acar, B. Yener, Unsupervised multiway data analysis: A literature survey, IEEE Transactions on Knowledge and Data Engineering.

[20] A. Cichocki, R. Zdunek, A.-H. Phan, S. Amari, Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation, John Wiley & Sons, Ltd, 2009.

[21] L. D. Lathauwer, B. D. Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl 21 (2000) 1253–1278.

[22] H. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, MPCA: Multilinear principal component analysis of tensor objects., IEEE Transactions on Neural Networks 19 (1) (2008) 18–39.

[23] H. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, A survey of multilinear subspace learning for tensor data, Pattern Recognition 44 (7) (2011) 1540 – 1551.

[24] C. F. Beckmann, S. M. Smith, Tensorial extensions of independent component analysis for multisubject FMRI analysis, Neuroimage 25 (1) (2005) 294–311.

[25] M. A. O. Vasilescu, D. Terzopoulos, Multilinear independent components analysis, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2005, pp. 547–553.

[26] M. Welling, M. Weber, Positive tensor factorization., Pattern Recognition Letters 22 (12) (2001) 1255–1261.

[27] A. Shashua, T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, in: Proceedings of the International Conference on Machine Learning, 2005, pp. 792–799.

[28] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, in: C. W. Harris (Ed.), Problems in Measuring Change, University of Wisconsin Press, 1963, pp. 122–137.

[29] M. Signoretto, L. D. Lathauwer, J. A. Suykens, A kernel-based framework to tensorial data analysis, Neural Networks 24 (8) (2011) 861 – 874, artificial Neural Networks: Selected Papers from ICANN 2010.

[30] Q. Zhao, G. Zhou, T. Adali, L. Zhang, A. Cichocki, Kernel-based tensor partial least squares for reconstruction of limb movements, in: ICASSP'13, 2013, pp. 3577–3581.

[31] C.-S. Lee, A. Elgammal, Modeling view and posture manifolds for tracking, IEEE International Conference on Computer Vision (2007) 1–8.

[32] X. Gao, C. Tian, Multi-view face recognition based on tensor subspace analysis and view manifold modeling, Neurocomput. 72 (16-18) (2009) 3742–3750.

[33] T. Furukawa, SOM of SOMs: Self-organizing map which maps a group of self-organizing maps, in: W. Duch, J. Kacprzyk, E. Oja, S. Zadrozny (Eds.), Artificial Neural Networks: Biological Inspirations, Vol. 3696 of Lecture Notes in Computer Science, Springer, 2005, pp. 391–396.

[34] T. Furukawa, SOM of SOMs, Neural Networks 22 (4) (2009) 463–478.

[35] T. Heskes, J.-J. Spanjers, W. Wiegerinck, EM algorithms for self-organizing maps., in: IJCNN (6), 2000, pp. 9–14.

[36] J. Verbeek, N. Vlassis, B. Krose, Self-organizing mixture models, Neurocomputing 63 (2005) 99–123. doi:10.1016/j.neucom.2004.04.008.

[37] S. P. Luttrell, Derivation of a class of training algorithms, IEEE Transactions on Neural Networks 1 (2) (1990) 229–232. doi:10.1109/72.80234.

[38] Y. Cheng, Convergence and ordering of kohonen's batch map, Neural Comput. 9 (8) (1997) 1667–1676.

[39] T. Graepel, M. Burger, K. Obermayer, Self-organizing maps: Generalizations and new optimization techniques, Neurocomputing 21 (1998) 173–190.

[40] I. Olier, A. Vellido, Variational bayesian generative topographic mapping., J. Math. Model. Algorithms 7 (4) (2008) 371–387.

[41] T. Kamishima, H. Kazawa, S. Akaho, A survey and empirical comparison of object ranking methods, in: J. Fürnkranz, E. Hüllermeier (Eds.), Preference Learning, Springer, 2010, pp. 181–201.

[42] A. Ultsch, H. P. Siemon, Kohonen's self organizing feature maps for exploratory data analysis., in: Proc. INNC'90, Int. Neural Network Conf., 1990, pp. 305–308.

[43] P. Stefanovic, O. Kurasova, Visual analysis of self-organzing maps, Nonlinear Analysis: Modelling and Control 16 (4) (2011) 488–504.

[44] B. Klimt, Y. Yang, The Enron corpus: A new dataset for email classification research, in: Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings, 2004, pp. 217–226.

[45] A. McCallum, X. Wang, A. Corrada-Emmanuel, Topic and role discovery in social networks with experiments on enron and academic email, J. Artif. Int. Res. 30 (1) (2007) 249–272.

[46] T. Ohkubo, T. Furukawa, K. Tokunaga, Requirements for the learning of multiple dynamics, in: J. Laaksonen, T. Honkela (Eds.), Advances in Self-Organizing Maps, Vol. 6731 of Lecture Notes in Computer Science, Springer, 2011, pp. 101–110.

[47] T. Furukawa, K. Natsume, T. Ohkubo, Research on multi-system learning theory: A case study of brain-inspired system research, in: Proc. of SCIS-ISIS 2012, 2012, pp. 311–314.

[48] P. Pajunen, A. Hyvarinen, J. Karhunen, Nonlinear blind source separation by self-organizing maps, in: In Proc. Int. Conf. on Neural Information Processing, 1996, pp. 1207–1210.

[49] M. Haritopoulos, H. Yin, N. M. Allinson, Image denoising using self-organizing map-based nonlinear independent component analysis., Neural Networks 15 (8-9) (2002) 1085–1098.

[50] S.-I. Amari, Topographic organization of nerve fields, Bulletin of Mathematical Biology 42 (3) (1980) 339–364. doi:10.1007/BF02460791.

[51] C. von der Malsburg, Self-organization of orientation sensitive cells in the striate cortex, Kybernetik 14 (2) (1973) 85–100. doi:10.1007/BF00288907.

[52] T. Kohonen, Self-organizing neural projections, Neural Networks 19 (6–7) (2006) 723 – 733.