Tensor Balancing on Statistical Manifold

Mahito Sugiyama [1 2] **Hiroyuki Nakahara** [3] **Koji Tsuda** [4 5 6]

Abstract

We solve *tensor balancing*, rescaling an Nth order nonnegative tensor by multiplying N tensors of order $N-1$ so that every fiber sums to one. This generalizes a fundamental process of *matrix balancing* used to compare matrices in a wide range of applications from biology to economics. We present an efficient balancing algorithm with quadratic convergence using Newton's method and show in numerical experiments that the proposed algorithm is several orders of magnitude faster than existing ones. To theoretically prove the correctness of the algorithm, we model tensors as probability distributions in a statistical manifold and realize tensor balancing as projection onto a submanifold. The key to our algorithm is that the gradient of the manifold, used as a Jacobian matrix in Newton's method, can be analytically obtained using the *Möbius inversion formula*, the essential of combinatorial mathematics. Our model is not limited to tensor balancing, but has a wide applicability as it includes various statistical and machine learning models such as weighted DAGs and Boltzmann machines.

1. Introduction

Matrix balancing is the problem of rescaling a given square nonnegative matrix $A \in \mathbb{R}^{n \times n}_{\geq 0}$ to a *doubly stochastic matrix* RAS, where every row and column sums to one, by multiplying two diagonal matrices R and S. This is a fundamental process for analyzing and comparing matrices in a wide range of applications, including input-output analysis in economics, called the RAS approach (Parikh, 1979; Miller & Blair, 2009; Lahr & de Mesnard, 2004), seat assignments in elections (Balinski, 2008; Akartunalı & Knight, 2016), Hi-C data analysis (Rao et al., 2014; Wu & Michor, 2016), the Sudoku puzzle (Moon et al., 2009), and the optimal transportation problem (Cuturi, 2013; Frogner et al., 2015; Solomon et al., 2015). An excellent review of this theory and its applications is given by Idel (2016).

[1]National Institute of Informatics [2]JST PRESTO [3]RIKEN Brain Science Institute [4]Graduate School of Frontier Sciences, The University of Tokyo [5]RIKEN AIP [6]NIMS. Correspondence to: Mahito Sugiyama <mahito@nii.ac.jp>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

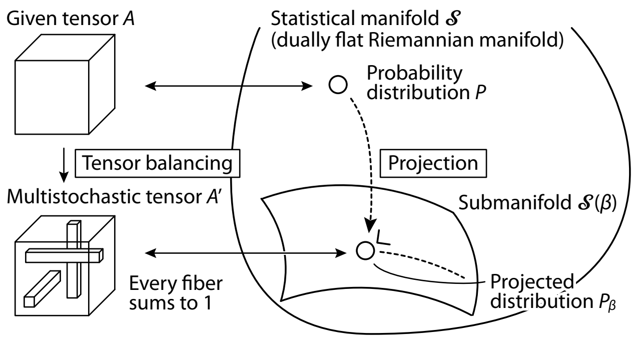

Given tensor A
Statistical manifold $\mathcal{S}$ (dually flat Riemannian manifold)
Probability distribution P
Tensor balancing
Projection
Multistochastic tensor A'
Submanifold $\mathcal{S}(\beta)$
Every fiber sums to 1
Projected distribution P_β

Figure 1. Overview of our approach.

The standard matrix balancing algorithm is the *Sinkhorn-Knopp algorithm* (Sinkhorn, 1964; Sinkhorn & Knopp, 1967; Marshall & Olkin, 1968; Knight, 2008), a special case of Bregman's balancing method (Lamond & Stewart, 1981) that iterates rescaling of each row and column until convergence. The algorithm is widely used in the above applications due to its simple implementation and theoretically guaranteed convergence. However, the algorithm converges linearly (Soules, 1991), which is prohibitively slow for recently emerging large and sparse matrices. Although Livne & Golub (2004) and Knight & Ruiz (2013) tried to achieve faster convergence by approximating each step of Newton's method, the exact Newton's method with quadratic convergence has not been intensively studied yet.

Another open problem is *tensor balancing*, which is a generalization of balancing from matrices to higher-order multidimentional arrays, or *tensors*. The task is to rescale an Nth order nonnegative tensor to a *multistochastic tensor*, in which every fiber sums to one, by multiplying $(N-1)$th order N tensors. There are some results about mathematical properties of multistochastic tensors (Cui et al., 2014; Chang et al., 2016; Ahmed et al., 2003). However, there is no result for tensor balancing algorithms with guaranteed convergence that transforms a given tensor to a multistochastic tensor until now.

Here we show that Newton's method with quadratic convergence can be applied to tensor balancing while avoiding solving a linear system on the full tensor. Our strategy is to realize matrix and tensor balancing as *projection onto a dually flat Riemmanian submanifold* (Figure 1), which is a statistical manifold and known to be the essential structure for probability distributions in information geometry (Amari, 2016). Using a partially ordered outcome space, we generalize the *log-linear model* (Agresti, 2012) used to model the higher-order combinations of binary variables (Amari, 2001; Ganmor et al., 2011; Nakahara & Amari, 2002; Nakahara et al., 2003), which allows us to model tensors as probability distributions in the statistical manifold. The remarkable property of our model is that the gradient of the manifold can be analytically computed using the *Möbius inversion formula* (Rota, 1964), the heart of combinatorial mathematics (Ito, 1993), which enables us to directly obtain the Jacobian matrix in Newton's method. Moreover, we show that $(n-1)^N$ entries for the size n^N of a tensor are invariant with respect to one of the two coordinate systems of the statistical manifold. Thus the number of equations in Newton's method is $O(n^{N-1})$.

The remainder of this paper is organized as follows: We begin with a low-level description of our matrix balancing algorithm in Section 2 and demonstrate its efficiency in numerical experiments in Section 3. To guarantee the correctness of the algorithm and extend it to tensor balancing, we provide theoretical analysis in Section 4. In Section 4.1, we introduce a generalized log-linear model associated with a partial order structured outcome space, followed by introducing the dually flat Riemannian structure in Section 4.2. In Section 4.3, we show how to use Newton's method to compute projection of a probability distribution onto a submanifold. Finally, we formulate the matrix and tensor balancing problem in Section 5 and summarize our contributions in Section 6.

2. The Matrix Balancing Algorithm

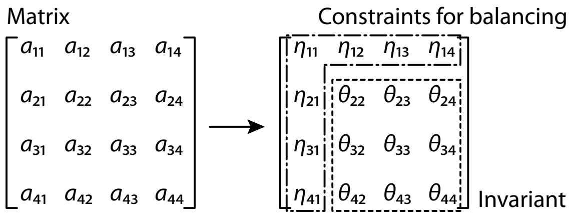
Given a nonnegative square matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}_{\geq 0}$, the task of *matrix balancing* is to find $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{R}^n$ that satisfy

$$(RAS)\mathbf{1} = \mathbf{1}, \quad (RAS)^T\mathbf{1} = \mathbf{1}, \qquad (1)$$

where $R = \mathrm{diag}(\boldsymbol{r})$ and $S = \mathrm{diag}(\boldsymbol{s})$. The balanced matrix $A' = RAS$ is called *doubly stochastic*, in which each entry $a'_{ij} = a_{ij}r_is_j$ and all the rows and columns sum to one. The most popular algorithm is the Sinkhorn-Knopp algorithm, which repeats updating $\boldsymbol{r}$ and $\boldsymbol{s}$ as $\boldsymbol{r} = 1/(A\boldsymbol{s})$ and $\boldsymbol{s} = 1/(A^T\boldsymbol{r})$. We denote by $[n] = \{1, 2, \dots, n\}$ hereafter.

In our algorithm, instead of directly updating $\boldsymbol{r}$ and $\boldsymbol{s}$, we update two parameters θ and η defined as

$$\log p_{ij} = \sum_{i'\leq i}\sum_{j'\leq j}\theta_{i'j'}, \quad \eta_{ij} = \sum_{i'\geq i}\sum_{j'\geq j}p_{i'j'} \qquad (2)$$



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \longrightarrow \begin{bmatrix} \eta_{11} & \eta_{12} & \eta_{13} & \eta_{14} \\ \eta_{21} & \theta_{22} & \theta_{23} & \theta_{24} \\ \eta_{31} & \theta_{32} & \theta_{33} & \theta_{34} \\ \eta_{41} & \theta_{42} & \theta_{43} & \theta_{44} \end{bmatrix}$$

Figure 2. Matrix balancing with two parameters θ and η.

for each $i, j \in [n]$, where we normalized entries as $p_{ij} = a_{ij}/\sum_{ij}a_{ij}$ so that $\sum_{ij}p_{ij} = 1$. We assume for simplicity that each entry is strictly larger than zero. The assumption will be removed in Section 5.

The key to our approach is that we update $\theta^{(t)}_{ij}$ with $i = 1$ or $j = 1$ by Newton's method at each iteration $t = 1, 2, \dots$ while fixing θ_{ij} with $i, j \neq 1$ so that $\eta^{(t)}_{ij}$ satisfies the following condition (Figure 2):

$$\eta^{(t)}_{i1} = (n - i + 1)/n, \quad \eta^{(t)}_{1j} = (n - j + 1)/n.$$

Note that the rows and columns sum not to 1 but to $1/n$ due to the normalization. The update formula is described as

$$\begin{bmatrix} \theta^{(t+1)}_{11} \\ \vdots \\ \theta^{(t+1)}_{1n} \\ \theta^{(t+1)}_{21} \\ \vdots \\ \theta^{(t+1)}_{n1} \end{bmatrix} = \begin{bmatrix} \theta^{(t)}_{11} \\ \vdots \\ \theta^{(t)}_{1n} \\ \theta^{(t)}_{21} \\ \vdots \\ \theta^{(t)}_{n1} \end{bmatrix} - J^{-1}\begin{bmatrix} \eta^{(t)}_{11} - (n-1+1)/n \\ \vdots \\ \eta^{(t)}_{1n} - (n-n+1)/n \\ \eta^{(t)}_{21} - (n-2+1)/n \\ \vdots \\ \eta^{(t)}_{n1} - (n-n+1)/n \end{bmatrix}, \qquad (3)$$

where J is the Jacobian matrix given as

$$J_{(ij)(i'j')} = \frac{\partial \eta^{(t)}_{ij}}{\partial \theta^{(t)}_{i'j'}} = \eta_{\max\{i,i'\}\max\{j,j'\}} - n^2\eta_{ij}\eta_{i'j'}, \qquad (4)$$

which is derived from our theoretical result in Theorem 3. Since J is a $(2n-1)\times(2n-1)$ matrix, the time complexity of each update is $O(n^3)$, which is needed to compute the inverse of J.

After updating to $\theta^{(t+1)}_{ij}$, we can compute $p^{(t+1)}_{ij}$ and $\eta^{(t+1)}_{ij}$ by Equation (2). Since this update does not ensure the condition $\sum_{ij}p^{(t+1)}_{ij} = 1$, we again update $\theta^{(t+1)}_{11}$ as $\theta^{(t+1)}_{11} = \theta^{(t+1)}_{11} - \log\sum_{ij}p^{(t+1)}_{ij}$ and recompute $p^{(t+1)}_{ij}$ and $\eta^{(t+1)}_{ij}$ for each $i, j \in [n]$.

By iterating the above update process in Equation (3) until convergence, $A = (a_{ij})$ with $a_{ij} = np_{ij}$ becomes doubly stochastic.

3. Numerical Experiments

We evaluate the efficiency of our algorithm compared to the two prominent balancing methods, the standard Sinkhorn-Knopp algorithm (Sinkhorn, 1964) and the state-of-the-art

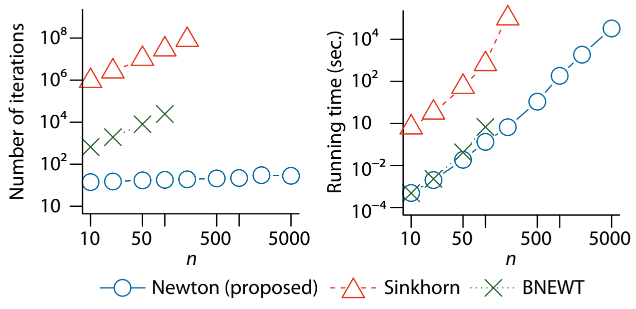


Figure 3. Results on Hessenberg matrices. The BNEWT algorithm (green) failed to converge for $n \geq 200$.

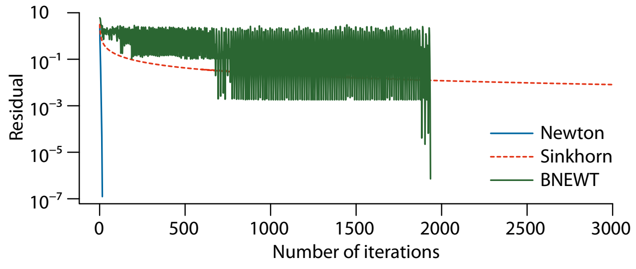


Figure 4. Convergence graph on H_{20}.

algorithm BNEWT (Knight & Ruiz, 2013), which uses Newton's method-like iterations with conjugate gradients. All experiments were conducted on Amazon Linux AMI release 2016.09 with a single core of 2.3 GHz Intel Xeon CPU E5-2686 v4 and 256 GB of memory. All methods were implemented in `C++` with the `Eigen` library and compiled with `gcc` 4.8.3[1]. We have carefully implemented BNEWT by directly translating the MATLAB code provided in (Knight & Ruiz, 2013) into `C++` with the `Eigen` library for fair comparison, and used the default parameters. We measured the residual of a matrix $A' = (a'_{ij})$ by the squared norm $\|(A'\mathbf{1} - \mathbf{1}, A'^T\mathbf{1} - \mathbf{1})\|_2$, where each entry a'_{ij} is obtained as np_{ij} in our algorithm, and ran each of three algorithms until the residual is below the tolerance threshold 10^{-6}.

Hessenberg Matrix. The first set of experiments used a Hessenberg matrix, which has been a standard benchmark for matrix balancing (Parlett & Landis, 1982; Knight & Ruiz, 2013). Each entry of an $n \times n$ Hessenberg matrix $H_n = (h_{ij})$ is given as $h_{ij} = 0$ if $j < i - 1$ and $h_{ij} = 1$ otherwise. We varied the size n from 10 to 5,000, and measured running time (in seconds) and the number of iterations of each method.

Results are plotted in Figure 3. Our balancing algorithm with the Newton's method (plotted in blue in the figures) is clearly the fastest: It is three to five orders of magnitude faster than the standard Sinkhorn-Knopp algorithm (plotted in red). Although the BNEWT algorithm (plotted in green) is competitive if n is small, it suddenly fails to converge whenever $n \geq 200$, which is consistent with results in the original paper (Knight & Ruiz, 2013) where there is no result for the setting $n \geq 200$ on the same matrix. Moreover, our method converges around 10 to 20 steps, which is about three and seven orders of magnitude smaller than BNEWT and Sinkhorn-Knopp, respectively, at $n = 100$.

To see the behavior of the rate of convergence in detail, we plot the convergence graph in Figure 4 for $n = 20$, where we observe the slow convergence rate of the Sinkhorn-Knopp algorithm and unstable convergence of the BNEWT algorithm, which contrasts with our quick convergence.

Trefethen Matrix. Next, we collected a set of Trefethen matrices from a collection website[2], which are nonnegative diagonal matrices with primes. Results are plotted in Figure 5, where we observe the same trend as before: Our algorithm is the fastest and about four orders of magnitude faster than the Sinkhorn-Knopp algorithm. Note that larger matrices with $n > 300$ do not have total support, which is the necessary condition for matrix balancing (Knight & Ruiz, 2013), while the BNEWT algorithm fails to converge if $n = 200$ or $n = 300$.

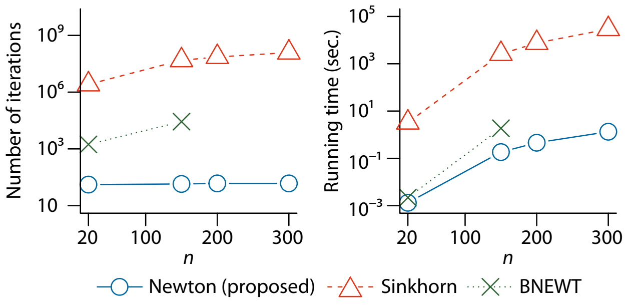


Figure 5. Results on Trefethen matrices. The BNEWT algorithm (green) failed to converge for $n \geq 200$.

4. Theoretical Analysis

In the following, we provide theoretical support to our algorithm by formulating the problem as a projection within a statistical manifold, in which a matrix corresponds to an element, that is, a probability distribution, in the manifold.

We show that a balanced matrix forms a submanifold and matrix balancing is projection of a given distribution onto the submanifold, where the Jacobian matrix in Equation (4) is derived from the gradient of the manifold.

[1]An implementation of algorithms for matrices and third order tensors is available at: `https://github.com/mahito-sugiyama/newton-balancing`

[2]`http://www.cise.ufl.edu/research/sparse/matrices/`

4.1. Formulation

We introduce our log-linear probabilistic model, where the outcome space is a partially ordered set, or a *poset* (Gierz et al., 2003). We prepare basic notations and the key mathematical tool for posets, the Möbius inversion formula, followed by formulating the log-linear model.

4.1.1. Möbius Inversion

A poset $(S, \leq)$, the set of elements S and a partial order $\leq$ on S, is a fundamental structured space in computer science. A *partial order* "$\leq$" is a relation between elements in S that satisfies the following three properties: For all $x, y, z \in S$, (1) $x \leq x$ (reflexivity), (2) $x \leq y$, $y \leq x \Rightarrow x = y$ (antisymmetry), and (3) $x \leq y$, $y \leq z \Rightarrow x \leq z$ (transitivity). In what follows, S is always finite and includes the least element (bottom) $\perp \in S$; that is, $\perp \leq x$ for all $x \in S$. We denote $S \setminus \{\perp\}$ by S^+.

Rota (1964) introduced the *Möbius inversion formula* on posets by generalizing the inclusion-exclusion principle. Let $\zeta \colon S \times S \to \{0, 1\}$ be the *zeta function* defined as

$$\zeta(s, x) = \begin{cases} 1 & \text{if } s \leq x, \\ 0 & \text{otherwise.} \end{cases}$$

The *Möbius function* $\mu \colon S \times S \to \mathbb{Z}$ satisfies $\zeta\mu = I$, which is inductively defined for all x, y with $x \leq y$ as

$$\mu(x, y) = \begin{cases} 1 & \text{if } x = y, \\ -\sum_{x \leq s < y} \mu(x, s) & \text{if } x < y, \\ 0 & \text{otherwise.} \end{cases}$$

From the definition, it follows that

$$\begin{aligned} \sum_{s \in S} \zeta(s, y)\mu(x, s) &= \sum_{x \leq s \leq y} \mu(x, s) = \delta_{xy}, \\ \sum_{s \in S} \zeta(x, s)\mu(s, y) &= \sum_{x \leq s \leq y} \mu(s, y) = \delta_{xy} \end{aligned} \qquad (5)$$

with the Kronecker delta δ such that $\delta_{xy} = 1$ if $x = y$ and $\delta_{xy} = 0$ otherwise. Then for any functions f, g, and h with the domain S such that

$$g(x) = \sum_{s \in S} \zeta(s, x) f(s) = \sum_{s \leq x} f(s),$$
$$h(x) = \sum_{s \in S} \zeta(x, s) f(s) = \sum_{s \geq x} f(s),$$

f is uniquely recovered with the Möbius function:

$$f(x) = \sum_{s \in S} \mu(s, x) g(s), \quad f(x) = \sum_{s \in S} \mu(x, s) h(s).$$

This is called the *Möbius inversion formula* and is at the heart of enumerative combinatorics (Ito, 1993).

4.1.2. Log-Linear Model on Posets

We consider a probability vector p on $(S, \leq)$ that gives a discrete probability distribution with the outcome space S. A probability vector is treated as a mapping $p \colon S \to (0, 1)$ such that $\sum_{x \in S} p(x) = 1$, where every entry $p(x)$ is assumed to be strictly larger than zero.

Using the zeta and the Möbius functions, let us introduce two mappings $\theta \colon S \to \mathbb{R}$ and $\eta \colon S \to \mathbb{R}$ as

$$\theta(x) = \sum_{s \in S} \mu(s, x) \log p(s), \qquad (6)$$

$$\eta(x) = \sum_{s \in S} \zeta(x, s) p(s) = \sum_{s \geq x} p(s). \qquad (7)$$

From the Möbius inversion formula, we have

$$\log p(x) = \sum_{s \in S} \zeta(s, x)\theta(s) = \sum_{s \leq x} \theta(s), \qquad (8)$$

$$p(x) = \sum_{s \in S} \mu(x, s)\eta(s). \qquad (9)$$

They are generalization of the *log-linear model* (Agresti, 2012) that gives the probability $p(\boldsymbol{x})$ of an n-dimensional binary vector $\boldsymbol{x} = (x^1, \dots, x^n) \in \{0, 1\}^n$ as

$$\log p(\boldsymbol{x}) = \sum_i \theta^i x^i + \sum_{i<j} \theta^{ij} x^i x^j + \sum_{i<j<k} \theta^{ijk} x^i x^j x^k + \dots + \theta^{1\dots n} x^1 x^2 \dots x^n - \psi,$$

where $\boldsymbol{\theta} = (\theta^1, \dots, \theta^{12\dots n})$ is a parameter vector, ψ is a normalizer, and $\boldsymbol{\eta} = (\eta^1, \dots, \eta^{12\dots n})$ represents the expectation of variable combinations such that

$$\begin{aligned} \eta^i &= \mathbf{E}[x^i] = \Pr(x^i = 1), \\ \eta^{ij} &= \mathbf{E}[x^i x^j] = \Pr(x^i = x^j = 1),\ i < j, \dots \\ \eta^{1\dots n} &= \mathbf{E}[x^1 \dots x^n] = \Pr(x^1 = \dots = x^n = 1). \end{aligned}$$

They coincide with Equations (8) and (7) when we let $S = 2^V$ with $V = \{1, 2, \dots, n\}$, each $x \in S$ as the set of indices of "1" of $\boldsymbol{x}$, and the order $\leq$ as the inclusion relationship, that is, $x \leq y$ if and only if $x \subseteq y$. Nakahara et al. (2006) have pointed out that $\boldsymbol{\theta}$ can be computed from p using the inclusion-exclusion principle in the log-linear model. We exploit this combinatorial property of the log-linear model using the Möbius inversion formula on posets and extend the log-linear model from the power set 2^V to any kind of posets $(S, \leq)$. Sugiyama et al. (2016) studied a relevant log-linear model, but the relationship with Möbius inversion formula has not been analyzed yet.

4.2. Dually Flat Riemannian Manifold

We theoretically analyze our log-linear model introduced in Equations (6), (7) and show that they form dual coordinate systems on a dually flat manifold, which has been mainly studied in the area of information geometry (Amari, 2001; Nakahara & Amari, 2002; Amari, 2014; 2016). Moreover, we show that the Riemannian metric and connection of our model can be analytically computed in closed forms.

In the following, we denote by ξ the function θ or η and by ∇ the gradient operator with respect to $S^+ = S \setminus \{\perp\}$, i.e., $(\nabla f(\xi))(x) = \partial f / \partial \xi(x)$ for $x \in S^+$, and denote by $\boldsymbol{\mathcal{S}}$ the set of probability distributions specified by probability vectors, which forms a statistical manifold. We use uppercase letters $P, Q, R, \dots$ for points (distributions) in $\boldsymbol{\mathcal{S}}$ and their lowercase letters $p, q, r, \dots$ for the corresponding probability vectors treated as mappings. We write θ_P and η_P if they are connected with p by Equations (6) and (7), respectively, and abbreviate subscripts if there is no ambiguity.

4.2.1. Dually Flat Structure

We show that $\boldsymbol{\mathcal{S}}$ has the *dually flat Riemannian structure* induced by two functions θ and η in Equation (6) and (7). We define $\psi(\theta)$ as

$$\psi(\theta) = -\theta(\perp) = -\log p(\perp), \tag{10}$$

which corresponds to the normalizer of p. It is a convex function since we have

$$\psi(\theta) = \log \sum_{x \in S} \exp\left(\sum_{\perp < s \leq x} \theta(s) \right)$$

from $\log p(x) = \sum_{\perp < s \leq x} \theta(s) - \psi(\theta)$. We apply the *Legendre transformation* to $\psi(\theta)$ given as

$$\varphi(\eta) = \max_{\theta'} \Big(\theta'\eta - \psi(\theta') \Big),\ \theta'\eta = \sum_{x \in S^+} \theta'(x)\eta(x). \tag{11}$$

Then $\varphi(\eta)$ coincides with the negative entropy.

Theorem 1 (Legendre dual)**.**

$$\varphi(\eta) = \sum_{x \in S} p(x) \log p(x).$$

Proof. From Equation (5), we have

$$\theta'\eta = \sum_{x \in S^+} \left(\sum_{\perp < s \leq x} \mu(s, x) \log p'(s) \sum_{s \geq x} p(s) \right)$$
$$= \sum_{x \in S^+} p(x) \left(\log p'(x) - \log p'(\perp) \right).$$

Thus it holds that

$$\theta'\eta - \psi(\theta') = \sum_{x \in S} p(x) \log p'(x). \tag{12}$$

Hence it is maximized with $p(x) = p'(x)$. □

Since they are connected with each other by the Legendre transformation, they form a *dual coordinate system* $\nabla\psi(\theta)$ and $\nabla\varphi(\eta)$ of $\boldsymbol{\mathcal{S}}$ (Amari, 2016, Section 1.5), which coincides with θ and η as follows.

Theorem 2 (dual coordinate system)**.**

$$\nabla\psi(\theta) = \eta, \quad \nabla\varphi(\eta) = \theta. \tag{13}$$

Proof. They can be directly derived from our definitions (Equations (6) and (11)) as

$$\frac{\partial \psi(\theta)}{\partial \theta(x)} = \frac{\sum_{y \geq x} \exp\Big(\sum_{\perp < s \leq y} \theta(s)\Big)}{\sum_{y \in S} \exp\Big(\sum_{\perp < s \leq y} \theta(s)\Big)} = \sum_{s \geq x} p(s) = \eta(x),$$
$$\frac{\partial \varphi(\eta)}{\partial \eta(x)} = \frac{\partial}{\partial \eta(x)} \Big(\theta\eta - \psi(\theta)\Big) = \theta(x). \qquad \square$$

Moreover, we can confirm the *orthogonality* of θ and η as

$$\mathbf{E}\left[\frac{\partial \log p(s)}{\partial \theta(x)} \frac{\partial \log p(s)}{\partial \eta(y)} \right] = \sum_{s \in S} \zeta(x, s)\mu(s, y) = \delta_{xy}.$$

The last equation holds from Equation (5), hence the Möbius inversion directly leads to the orthogonality.

The *Bregman divergence* is known to be the canonical divergence (Amari, 2016, Section 6.6) to measure the difference between two distributions P and Q on a dually flat manifold, which is defined as

$$D\left[P, Q\right] = \psi(\theta_P) + \varphi(\eta_Q) - \theta_P \eta_Q.$$

In our case, since we have $\varphi(\eta_Q) = \sum_{x \in S} q(x) \log q(x)$ and $\theta_P \eta_Q - \psi(\theta_P) = \sum_{x \in S} q(x) \log p(x)$ from Theorem 1 and Equation (12), it is given as

$$D\left[P, Q\right] = \sum_{x \in S} q(x) \log \frac{q(x)}{p(x)},$$

which coincides with the *Kullback–Leibler divergence* (KL divergence) from Q to P: $D\left[P, Q\right] = D_{\mathrm{KL}}\left[Q, P\right]$.

4.2.2. Riemannian Structure

Next we analyze the Riemannian structure on $\boldsymbol{\mathcal{S}}$ and show that the Möbius inversion formula enables us to compute the Riemannian metric of $\boldsymbol{\mathcal{S}}$.

Theorem 3 (Riemannian metric)**.** *The manifold $(\boldsymbol{\mathcal{S}}, g(\xi))$ is a Riemannian manifold with the Riemannian metric $g(\xi)$ such that for all $x, y \in S^+$*

$$g_{xy}(\xi) = \begin{cases} \sum_{s \in S} \Big[\zeta(x, s)\zeta(y, s)p(s) - \eta(x)\eta(y) \Big] & \text{if } \xi = \theta, \\ \sum_{s \in S} \mu(s, x)\mu(s, y)p(s)^{-1} & \text{if } \xi = \eta. \end{cases}$$

Proof. Since the Riemannian metric is defined as

$$g(\theta) = \nabla\nabla\psi(\theta), \quad g(\eta) = \nabla\nabla\varphi(\eta),$$

when $\xi = \theta$ we have

$$g_{xy}(\theta) = \frac{\partial^2}{\partial \theta(x) \partial \theta(y)} \psi(\theta) = \frac{\partial}{\partial \theta(x)} \eta(y)$$
$$= \frac{\partial}{\partial \theta(x)} \sum_{s \in S} \zeta(y, s) \exp\left(\sum_{\perp < u \leq s} \theta(u) - \psi(\theta) \right)$$

$$= \sum_{s \in S} \zeta(x, s)\zeta(y, s)p(s) - |S|\eta(x)\eta(y).$$

When $\xi = \eta$, it follows that

$$\begin{aligned} g_{xy}(\eta) &= \frac{\partial^2}{\partial\eta(x)\partial\eta(y)}\varphi(\eta) = \frac{\partial}{\partial\eta(x)}\theta(y) \\ &= \frac{\partial}{\partial\eta(x)} \sum_{s \le y} \mu(s, y) \log \left(\sum_{u \ge s} \mu(s, u)\eta(u) \right) \\ &= \sum_{s \in S} \mu(s, x)\mu(s, y)p(s)^{-1}. \end{aligned}$$

□

Since $g(\xi)$ coincides with the Fisher information matrix,

$$\mathbf{E}\left[\frac{\partial}{\partial\theta(x)} \log p(s) \frac{\partial}{\partial\theta(y)} \log p(s) \right] = g_{xy}(\theta),$$
$$\mathbf{E}\left[\frac{\partial}{\partial\eta(x)} \log p(s) \frac{\partial}{\partial\eta(y)} \log p(s) \right] = g_{xy}(\eta).$$

Then the Riemannian (Levi–Chivita) connection $\Gamma(\xi)$ with respect to ξ, which is defined as

$$\Gamma_{xyz}(\xi) = \frac{1}{2}\left(\frac{\partial g_{yz}(\xi)}{\partial\xi(x)} + \frac{\partial g_{xz}(\xi)}{\partial\xi(y)} - \frac{\partial g_{xy}(\xi)}{\partial\xi(z)} \right)$$

for all $x, y, z \in S^+$, can be analytically obtained.

Theorem 4 (Riemannian connection)**.** *The Riemannian connection $\Gamma(\xi)$ on the manifold $(\boldsymbol{\mathcal{S}}, g(\xi))$ is given in the following for all $x, y, z \in S^+$,*

$$\Gamma_{xyz}(\xi) = \begin{cases} \frac{1}{2} \sum_{s \in S} \big(\zeta(x, s) - \eta(x)\big)\big(\zeta(y, s) - \eta(y)\big) \\ \qquad \big(\zeta(z, s) - \eta(z)\big)p(s) & \text{if } \xi = \theta, \\ -\frac{1}{2} \sum_{s \in S} \mu(s, x)\mu(s, y)\mu(s, z)p(s)^{-2} & \text{if } \xi = \eta. \end{cases}$$

Proof. Connections $\Gamma_{xyz}(\theta)$ and $\Gamma_{xyz}(\eta)$ can be obtained by directly computing $\partial g_{yz}(\theta)/\partial\theta(x)$ and $\partial g_{yz}(\eta)/\partial\eta(x)$, respectively. □

4.3. The Projection Algorithm

Projection of a distribution onto a submanifold is essential; several machine learning algorithms are known to be formulated as projection of a distribution empirically estimated from data onto a submanifold that is specified by the target model (Amari, 2016). Here we define projection of distributions on posets and show that Newton's method can be applied to perform projection as the Jacobian matrix can be analytically computed.

4.3.1. Definition

Let $\boldsymbol{\mathcal{S}}(\beta)$ be a submanifold of $\boldsymbol{\mathcal{S}}$ such that

$$\boldsymbol{\mathcal{S}}(\beta) = \{ P \in \boldsymbol{\mathcal{S}} \mid \theta_P(x) = \beta(x),\ \forall x \in \mathrm{dom}(\beta) \} \quad (14)$$

specified by a function β with $\mathrm{dom}(\beta) \subseteq S^+$. Projection of $P \in \boldsymbol{\mathcal{S}}$ onto $\boldsymbol{\mathcal{S}}(\beta)$, called *$m$-projection*, which is defined as the distribution $P_\beta \in \boldsymbol{\mathcal{S}}(\beta)$ such that

$$\begin{cases} \theta_{P_\beta}(x) = \beta(x) & \text{if } x \in \mathrm{dom}(\beta), \\ \eta_{P_\beta}(x) = \eta_P(x) & \text{if } x \in S^+ \setminus \mathrm{dom}(\beta), \end{cases}$$

is the minimizer of the KL divergence from P to $\boldsymbol{\mathcal{S}}(\beta)$:

$$P_\beta = \operatorname*{argmin}_{Q \in \boldsymbol{\mathcal{S}}(\beta)} D_{\mathrm{KL}}[P, Q].$$

The dually flat structure with the coordinate systems θ and η guarantees that the projected distribution P_β always exists and is unique (Amari, 2009, Theorem 3). Moreover, the *Pythagorean theorem* holds in the dually flat manifold, that is, for any $Q \in \boldsymbol{\mathcal{S}}(\beta)$ we have

$$D_{\mathrm{KL}}[P, Q] = D_{\mathrm{KL}}[P, P_\beta] + D_{\mathrm{KL}}[P_\beta, Q].$$

We can switch η and θ in the submanifold $\boldsymbol{\mathcal{S}}(\beta)$ by changing $D_{\mathrm{KL}}[P, Q]$ to $D_{\mathrm{KL}}[Q, P]$, where the projected distribution P_β of P is given as

$$\begin{cases} \theta_{P_\beta}(x) = \theta_P(x) & \text{if } x \in S^+ \setminus \mathrm{dom}(\beta), \\ \eta_{P_\beta}(x) = \beta(x) & \text{if } x \in \mathrm{dom}(\beta), \end{cases}$$

This projection is called *e-projection*.

Example 1 (Boltzmann machine)**.** Given a Boltzmann machine represented as an undirected graph $G = (V, E)$ with a vertex set V and an edge set $E \subseteq \{\{i, j\} \mid i, j \in V\}$. The set of probability distributions that can be modeled by a Boltzmann machine G coincides with the submanifold

$$\boldsymbol{\mathcal{S}}_{\mathrm{B}} = \{ P \in \boldsymbol{\mathcal{S}} \mid \theta_P(x) = 0 \text{ if } |x| > 2 \text{ or } x \notin E \},$$

with $S = 2^V$. Let $\hat{P}$ be an empirical distribution estimated from a given dataset. The learned model is the m-projection of the empirical distribution $\hat{P}$ onto $\boldsymbol{\mathcal{S}}_{\mathrm{B}}$, where the resulting distribution P_β is given as

$$\begin{cases} \theta_{P_\beta}(x) = 0 & \text{if } |x| > 2 \text{ or } x \notin E, \\ \eta_{P_\beta}(x) = \eta_{\hat{P}}(x) & \text{if } |x| = 1 \text{ or } x \in E. \end{cases}$$

4.3.2. Computation

Here we show how to compute projection of a given probability distribution. We show that Newton's method can be used to efficiently compute the projected distribution P_β by iteratively updating $P_\beta^{(0)} = P$ as $P_\beta^{(0)}, P_\beta^{(1)}, P_\beta^{(2)}, \dots$ until converging to P_β.

Let us start with the m-projection with initializing $P_\beta^{(0)} = P$. In each iteration t, we update $\theta_{P_\beta}^{(t)}(x)$ for all $x \in \mathrm{dom}\beta$ while fixing $\eta_{P_\beta}^{(t)}(x) = \eta_P(x)$ for all $x \in S^+ \setminus \mathrm{dom}(\beta)$, which is possible from the orthogonality of θ and η. Using Newton's method, $\eta_{P_\beta}^{(t+1)}(x)$ should satisfy

$$\left(\theta_{P_\beta}^{(t)}(x) - \beta(x) \right) + \sum_{y \in \mathrm{dom}(\beta)} J_{xy} \left(\eta_{P_\beta}^{(t+1)}(y) - \eta_{P_\beta}^{(t)}(y) \right) = 0,$$

for every $x \in \mathrm{dom}(\beta)$, where J_{xy} is an entry of the $|\mathrm{dom}(\beta)| \times |\mathrm{dom}(\beta)|$ Jacobian matrix J and given as

$$J_{xy} = \frac{\partial \theta^{(t)}_{P_\beta}(x)}{\partial \eta^{(t)}_{P_\beta}(y)} = \sum_{s \in S} \mu(s,x)\mu(s,y)p^{(t)}_\beta(s)^{-1}$$

from Theorem 3. Therefore, we have the update formula for all $x \in \mathrm{dom}(\beta)$ as

$$\eta^{(t+1)}_{P_\beta}(x) = \eta^{(t)}_{P_\beta}(x) - \sum_{y \in \mathrm{dom}(\beta)} J^{-1}_{xy} \left(\theta^{(t)}_{P_\beta}(y) - \beta(y) \right).$$

In e-projection, update $\eta^{(t)}_{P_\beta}(x)$ for $x \in \mathrm{dom}(\beta)$ while fixing $\theta^{(t)}_{P_\beta}(x) = \theta_P(x)$ for all $x \in S^+ \setminus \mathrm{dom}(\beta)$. To ensure $\eta^{(t)}_{P_\beta}(\perp) = 1$, we add $\perp$ to $\mathrm{dom}(\beta)$ and $\beta(\perp) = 1$. We update $\theta^{(t)}_{P_\beta}(x)$ at each step t as

$$\theta^{(t+1)}_{P_\beta}(x) = \theta^{(t)}_{P_\beta}(x) - \sum_{y \in \mathrm{dom}(\beta)} {J'}^{-1}_{xy} \left(\eta^{(t)}_{P_\beta}(y) - \beta(y) \right),$$

$$J'_{xy} = \frac{\partial \eta^{(t)}_{P_\beta}(x)}{\partial \theta^{(t)}_{P_\beta}(y)} = \sum_{s \in S} \zeta(x,s)\zeta(y,s)p^{(t)}_\beta(s) - |S|\eta^{(t)}_{P_\beta}(x)\eta^{(t)}_{P_\beta}(y).$$

In this case, we also need to update $\theta^{(t)}_{P_\beta}(\perp)$ as it is not guaranteed to be fixed. Let us define

$$p'^{(t+1)}_\beta(x) = p^{(t)}_\beta(x) \prod_{s \in \mathrm{dom}(\beta)} \frac{\exp\left(\theta^{(t+1)}_{P_\beta}(s)\right)}{\exp\left(\theta^{(t)}_{P_\beta}(s)\right)} \zeta(s,x).$$

Since we have

$$p^{(t+1)}_\beta(x) = \frac{\exp\left(\theta^{(t+1)}_{P_\beta}(\perp)\right)}{\exp\left(\theta^{(t)}_{P_\beta}(\perp)\right)} p'^{(t+1)}_\beta(x),$$

it follows that

$$\begin{aligned} &\theta^{(t+1)}_{P_\beta}(\perp) - \theta^{(t)}_{P_\beta}(\perp) \\ &= -\log\left(\exp\left(\theta^{(t)}_{P_\beta}(\perp)\right) + \sum_{x \in S^+} p'^{(t+1)}_\beta(x) \right), \end{aligned}$$

The time complexity of each iteration is $O(|\mathrm{dom}(\beta)|^3)$, which is required to compute the inverse of the Jacobian matrix.

Global convergence of the projection algorithm is always guaranteed by the convexity of a submanifold $\boldsymbol{\mathcal{S}}(\beta)$ defined in Equation (14). Since $\boldsymbol{\mathcal{S}}(\beta)$ is always convex with respect to the θ- and η-coordinates, it is straightforward to see that our e-projection is an instance of the *Bregman algorithm* onto a convex region, which is well known to always converge to the global solution (Censor & Lent, 1981).

5. Balancing Matrices and Tensors

Now we are ready to solve the problem of matrix and tensor balancing as projection on a dually flat manifold.

5.1. Matrix Balancing

Recall that the task of matrix balancing is to find $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{R}^n$ that satisfy $(RAS)\mathbf{1} = \mathbf{1}$ and $(RAS)^T\mathbf{1} = \mathbf{1}$ with $R = \mathrm{diag}(\boldsymbol{r})$ and $S = \mathrm{diag}(\boldsymbol{s})$ for a given nonnegative square matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}_{\geq 0}$.

Let us define S as

$$S = \{(i,j) \mid i,j \in [n] \text{ and } a_{ij} \neq 0\}, \tag{15}$$

where we remove zero entries from the outcome space S as our formulation cannot treat zero probability, and give each probability as $p((i,j)) = a_{ij} / \sum_{ij} a_{ij}$. The partial order $\leq$ of S is naturally introduced as

$$x = (i,j) \leq y = (k,l) \Leftrightarrow i \leq j \text{ and } k \leq l, \tag{16}$$

resulting in $\perp = (1,1)$. In addition, we define $\boldsymbol{\iota}_{k,m}$ for each $k \in [n]$ and $m \in \{1,2\}$ such that

$$\boldsymbol{\iota}_{k,m} = \min\{\, x = (i_1, i_2) \in S \mid i_m = k \,\},$$

where the minimum is with respect to the order $\leq$. If $\boldsymbol{\iota}_{k,m}$ does not exist, we just remove the entire kth row if $m = 1$ or kth column if $m = 2$ from A. Then we switch rows and columns of A so that the condition

$$\boldsymbol{\iota}_{1,m} \leq \boldsymbol{\iota}_{2,m} \leq \dots \leq \boldsymbol{\iota}_{n,m} \tag{17}$$

is satisfied for each $m \in \{1,2\}$, which is possible for any matrices. Since we have

$$\eta(\boldsymbol{\iota}_{k,m}) - \eta(\boldsymbol{\iota}_{k+1,m}) = \begin{cases} \sum_{j=1}^n p((k,j)) & \text{if } m = 1, \\ \sum_{i=1}^n p((i,k)) & \text{if } m = 2 \end{cases}$$

if the condition (17) is satisfied, the probability distribution is balanced if for all $k \in [n]$ and $m \in \{1,2\}$

$$\eta(\boldsymbol{\iota}_{k,m}) = \frac{n-k+1}{n}.$$

Therefore, we obtain the following result.

Matrix balancing as *e*-projection:
Given a matrix $A \in \mathbb{R}^{n \times n}$ with its normalized probability distribution $P \in \boldsymbol{\mathcal{S}}$ such that $p((i,j)) = a_{ij} / \sum_{ij} a_{ij}$. Define the poset $(S, \leq)$ by Equations (15) and (16) and let $\boldsymbol{\mathcal{S}}(\beta)$ be the submanifold of $\boldsymbol{\mathcal{S}}$ such that

$$\boldsymbol{\mathcal{S}}(\beta) = \{P \in \boldsymbol{\mathcal{S}} \mid \eta_P(x) = \beta(x) \text{ for all } x \in \mathrm{dom}(\beta)\},$$

where the function β is given as

$$\mathrm{dom}(\beta) = \{\boldsymbol{\iota}_{k,m} \in S \mid k \in [n], m \in \{1,2\}\},$$

$$\beta(\boldsymbol{\iota}_{k,m}) = \frac{n-k+1}{n}.$$

Matrix balancing is the e-projection of P onto the submanifold $\boldsymbol{\mathcal{S}}(\beta)$, that is, the balanced matrix $(RAS)/n$ is the distribution P_β such that

$$\begin{cases} \theta_{P_\beta}(x) = \theta_P(x) & \text{if } x \in S^+ \setminus \mathrm{dom}(\beta), \\ \eta_{P_\beta}(x) = \beta(x) & \text{if } x \in \mathrm{dom}(\beta), \end{cases}$$

which is unique and always exists in $\boldsymbol{\mathcal{S}}$, thanks to its dually flat structure. Moreover, two balancing vectors $\boldsymbol{r}$ and $\boldsymbol{s}$ are

$$\exp\left(\sum_{k=1}^{i} \theta_{P_\beta}(\boldsymbol{\iota}_{k,m}) - \theta_P(\boldsymbol{\iota}_{k,m}) \right) = \begin{cases} r_i & \text{if } m = 1, \\ a_i & \text{if } m = 2, \end{cases}$$

for every $i \in [n]$ and $\boldsymbol{r} = \boldsymbol{r}n / \sum_{ij} a_{ij}$. ■

5.2. Tensor Balancing

Next, we generalize our approach from matrices to tensors. For an Nth order tensor $A = (a_{i_1 i_2 \dots i_N}) \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_N}$ and a vector $\boldsymbol{b} \in \mathbb{R}^{n_m}$, the m-mode product of A and $\boldsymbol{b}$ is defined as

$$(A \times_m \boldsymbol{b})_{i_1 \dots i_{m-1} i_{m+1} \dots i_N} = \sum_{i_m=1}^{n_m} a_{i_1 i_2 \dots i_N} b_{i_m}.$$

We define *tensor balancing* as follows: Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_N}$ with $n_1 = \dots = n_N = n$, find $(N-1)$ order tensors $R^1, R^2, \dots, R^N$ such that

$$A' \times_m \mathbf{1} = \mathbf{1} \; (\in \mathbb{R}^{n_1 \times \dots \times n_{m-1} \times n_{m+1} \times \dots \times n_N}) \qquad (18)$$

for all $m \in [N]$, i.e., $\sum_{i_m=1}^{n} a'_{i_1 i_2 \dots i_N} = 1$, where each entry $a'_{i_1 i_2 \dots i_N}$ of the balanced tensor A' is given as

$$a'_{i_1 i_2 \dots i_N} = a_{i_1 i_2 \dots i_N} \prod_{m \in [N]} R^m_{i_1 \dots i_{m-1} i_{m+1} \dots i_N}.$$

A tensor A' that satisfies Equation (18) is called *multistochastic* (Cui et al., 2014). Note that this is exactly the same as the matrix balancing problem if $N = 2$.

It is straightforward to extend matrix balancing to tensor balancing as e-projection onto a submanifold. Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_N}$ with its normalized probability distribution P such that

$$p(x) = a_{i_1 i_2 \dots i_N} \Big/ \sum_{j_1 j_2 \dots j_N} a_{j_1 j_2 \dots j_N} \qquad (19)$$

for all $x = (i_1, i_2, \dots, i_N)$. The objective is to obtain P_β such that $\sum_{i_m=1}^{n} p_\beta((i_1, \dots, i_N)) = 1/(n^{N-1})$ for all $m \in [N]$ and $i_1, \dots, i_N \in [n]$. In the same way as matrix balancing, we define S as

$$S = \left\{ (i_1, i_2, \dots, i_N) \in [n]^N \;\middle|\; a_{i_1 i_2 \dots i_N} \neq 0 \right\}$$

with removing zero entries and the partial order $\leq$ as

$$x = (i_1 \dots i_N) \leq y = (j_1 \dots j_N) \Leftrightarrow \forall m \in [N], i_m \leq j_m.$$

In addition, we introduce $\boldsymbol{\iota}_{k,m}$ as

$$\boldsymbol{\iota}_{k,m} = \min\{\, x = (i_1, i_2, \dots, i_N) \in S \mid i_m = k \,\}.$$

and require the condition in Equation (17).

Tensor balancing as *e*-projection:
Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_N}$ with its normalized probability distribution $P \in \boldsymbol{\mathcal{S}}$ given in Equation (19). The submanifold $\boldsymbol{\mathcal{S}}(\beta)$ of multistochastic tensors is given as

$$\boldsymbol{\mathcal{S}}(\beta) = \{ P \in \boldsymbol{\mathcal{S}} \mid \eta_P(x) = \beta(x) \text{ for all } x \in \mathrm{dom}(\beta) \},$$

where the domain of the function β is given as

$$\mathrm{dom}(\beta) = \{\, \boldsymbol{\iota}_{k,m} \mid k \in [n], m \in [N] \,\}$$

and each value is described using the zeta function as

$$\beta(\boldsymbol{\iota}_{k,m}) = \sum_{l \in [n]} \zeta(\boldsymbol{\iota}_{k,m}, \boldsymbol{\iota}_{l,m}) \frac{1}{n^{N-1}}.$$

Tensor balancing is the e-projection of P onto the submanifold $\boldsymbol{\mathcal{S}}(\beta)$, that is, the multistochastic tensor is the distribution P_β such that

$$\begin{cases} \theta_{P_\beta}(x) = \theta_P(x) & \text{if } x \in S^+ \setminus \mathrm{dom}(\beta), \\ \eta_{P_\beta}(x) = \beta(x) & \text{if } x \in \mathrm{dom}(\beta), \end{cases}$$

which is unique and always exists in $\boldsymbol{\mathcal{S}}$, thanks to its dually flat structure. Moreover, each balancing tensor R^m is

$$R^m_{i_1 \dots i_{m-1} i_{m+1} \dots i_N} = \exp\left(\sum_{m' \neq m} \sum_{k=1}^{i_{m'}} \theta_{P_\beta}(\boldsymbol{\iota}_{k,m'}) - \theta_P(\boldsymbol{\iota}_{k,m'}) \right)$$

for every $m \in [N]$ and $R^1 = R^1 n^{N-1} / \sum_{j_1 \dots j_N} a_{j_1 \dots j_N}$ to recover a multistochastic tensor. ■

Our result means that the e-projection algorithm based on Newton's method proposed in Section 4.3 converges to the unique balanced tensor whenever $\boldsymbol{\mathcal{S}}(\beta) \neq \emptyset$ holds.

6. Conclusion

In this paper, we have solved the open problem of tensor balancing and presented an efficient balancing algorithm using Newton's method. Our algorithm quadratically converges, while the popular Sinkhorn-Knopp algorithm linearly converges. We have examined the efficiency of our algorithm in numerical experiments on matrix balancing and showed that the proposed algorithm is several orders of magnitude faster than the existing approaches.

We have analyzed theories behind the algorithm, and proved that balancing is e-projection in a special type of a statistical manifold, in particular, a dually flat Riemannian manifold studied in information geometry. Our key finding is that the gradient of the manifold, equivalent to Riemannian metric or the Fisher information matrix, can be analytically obtained using the Möbius inversion formula.

Our information geometric formulation can model several machine learning applications such as statistical analysis on a DAG structure. Thus, we can perform efficient learning as projection using information of the gradient of manifolds by reformulating such models, which we will study in future work.

Acknowledgements

The authors sincerely thank Marco Cuturi for his valuable comments. This work was supported by JSPS KAKENHI Grant Numbers JP16K16115, JP16H02870 (MS), JP26120732 and JP16H06570 (HN). The research of K.T. was supported by JST CREST JPMJCR1502, RIKEN PostK, KAKENHI Nanostructure and KAKENHI JP15H05711.

References

Agresti, A. *Categorical data analysis*. Wiley, 3 edition, 2012.

Ahmed, M., De Loera, J., and Hemmecke, R. *Polyhedral Cones of Magic Cubes and Squares*, volume 25 of *Algorithms and Combinatorics*, pp. 25–41. Springer, 2003.

Akartunalı, K. and Knight, P. A. Network models and biproportional rounding for fair seat allocations in the UK elections. *Annals of Operations Research*, pp. 1–19, 2016.

Amari, S. Information geometry on hierarchy of probability distributions. *IEEE Transactions on Information Theory*, 47(5):1701–1711, 2001.

Amari, S. Information geometry and its applications: Convex function and dually flat manifold. In Nielsen, F. (ed.), *Emerging Trends in Visual Computing: LIX Fall Colloquium, ETVC 2008, Revised Invited Papers*, pp. 75–102. Springer, 2009.

Amari, S. Information geometry of positive measures and positive-definite matrices: Decomposable dually flat structure. *Entropy*, 16(4):2131–2145, 2014.

Amari, S. *Information Geometry and Its Applications*. Springer, 2016.

Balinski, M. Fair majority voting (or how to eliminate gerrymandering). *American Mathematical Monthly*, 115(2): 97–113, 2008.

Censor, Y. and Lent, A. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, 1981.

Chang, H., Paksoy, V. E., and Zhang, F. Polytopes of stochastic tensors. *Annals of Functional Analysis*, 7(3): 386–393, 2016.

Cui, L.-B., Li, W., and Ng, M. K. Birkhoff–von Neumann theorem for multistochastic tensors. *SIAM Journal on Matrix Analysis and Applications*, 35(3):956–973, 2014.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26*, pp. 2292–2300, 2013.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T. A. Learning with a Wasserstein loss. In *Advances in Neural Information Processing Systems 28*, pp. 2053–2061, 2015.

Ganmor, E., Segev, R., and Schneidman, E. Sparse low-order interaction network underlies a highly correlated and learnable neural population code. *Proceedings of the National Academy of Sciences*, 108(23):9679–9684, 2011.

Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J. D., Mislove, M., and Scott, D. S. *Continuous Lattices and Domains*. Cambridge University Press, 2003.

Idel, M. A review of matrix scaling and sinkhorn's normal form for matrices and positive maps. *arXiv:1609.06349*, 2016.

Ito, K. (ed.). *Encyclopedic Dictionary of Mathematics*. The MIT Press, 2 edition, 1993.

Knight, P. A. The Sinkhorn–Knopp algorithm: Convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

Knight, P. A. and Ruiz, D. A fast algorithm for matrix balancing. *IMA Journal of Numerical Analysis*, 33(3): 1029–1047, 2013.

Lahr, M. and de Mesnard, L. Biproportional techniques in input-output analysis: Table updating and structural analysis. *Economic Systems Research*, 16(2):115–134, 2004.

Lamond, B. and Stewart, N. F. Bregman's balancing method. *Transportation Research Part B: Methodological*, 15(4):239–248, 1981.

Livne, O. E. and Golub, G. H. Scaling by binormalization. *Numerical Algorithms*, 35(1):97–120, 2004.

Marshall, A. W. and Olkin, I. Scaling of matrices to achieve specified row and column sums. *Numerische Mathematik*, 12(1):83–90, 1968.

Miller, R. E. and Blair, P. D. *Input-Output Analysis: Foundations and Extensions*. Cambridge University Press, 2 edition, 2009.

Moon, T. K., Gunther, J. H., and Kupin, J. J. Sinkhorn solves sudoku. *IEEE Transactions on Information Theory*, 55(4):1741–1746, 2009.

Nakahara, H. and Amari, S. Information-geometric measure for neural spikes. *Neural Computation*, 14(10): 2269–2316, 2002.

Nakahara, H., Nishimura, S., Inoue, M., Hori, G., and Amari, S. Gene interaction in DNA microarray data is decomposed by information geometric measure. *Bioinformatics*, 19(9):1124–1131, 2003.

Nakahara, H., Amari, S., and Richmond, B. J. A comparison of descriptive models of a single spike train by information-geometric measure. *Neural computation*, 18 (3):545–568, 2006.

Parikh, A. Forecasts of input-output matrices using the R.A.S. method. *The Review of Economics and Statistics*, 61(3):477–481, 1979.

Parlett, B. N. and Landis, T. L. Methods for scaling to doubly stochastic form. *Linear Algebra and its Applications*, 48:53–79, 1982.

Rao, S. S. P., Huntley, M. H., Durand, N. C., Stamenova, E. K., Bochkov, I. D., Robinson, J. T., Sanborn, A. L., Machol, I., Omer, A. D., Lander, E. S., and Aiden, E. L. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7): 1665–1680, 2014.

Rota, G.-C. On the foundations of combinatorial theory I: Theory of Möbius functions. *Z. Wahrseheinlichkeitstheorie*, 2:340–368, 1964.

Sinkhorn, R. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 06 1964.

Sinkhorn, R. and Knopp, P. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Solomon, J., de Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66:1–66:11, 2015.

Soules, G. W. The rate of convergence of sinkhorn balancing. *Linear Algebra and its Applications*, 150:3–40, 1991.

Sugiyama, M., Nakahara, H., and Tsuda, K. Information decomposition on structured space. In *2016 IEEE International Symposium on Information Theory*, pp. 575–579, July 2016.

Wu, H.-J. and Michor, F. A computational strategy to adjust for copy number in tumor Hi-C data. *Bioinformatics*, 32 (24):3695–3701, 2016.