Structured BFGS Method for Optimal Doubly Stochastic Matrix Approximation

Dejun Chu[1], **Changshui Zhang**[2], **Shiliang Sun**[3], **Qing Tao**[4]
[1] School of Software, Hefei University of Technology
[2] Department of Automation, Tsinghua University
[3] School of Computer Science and Technology, East China Normal University
[4] Army Academy of Artillery and Air Defense
djun.chu@gmail.com, zcs@mail.tsinghua.edu.cn, slsun@cs.ecnu.edu.cn, taoqing@gmail.com

Abstract

Doubly stochastic matrix plays an essential role in several areas such as statistics and machine learning. In this paper we consider the optimal approximation of a square matrix in the set of doubly stochastic matrices. A structured BFGS method is proposed to solve the dual of the primal problem. The resulting algorithm builds curvature information into the diagonal components of the true Hessian, so that it takes only additional linear cost to obtain the descent direction based on the gradient information without having to explicitly store the inverse Hessian approximation. The cost is substantially fewer than quadratic complexity of the classical BFGS algorithm. Meanwhile, a Newton-based line search method is presented for finding a suitable step size, which in practice uses the existing knowledge and takes only one iteration. The global convergence of our algorithm is established. We verify the advantages of our approach on both synthetic data and real data sets. The experimental results demonstrate that our algorithm outperforms the state-of-the-art solvers and enjoys outstanding scalability.

Introduction

A square matrix X is called a doubly stochastic matrix if it is nonnegative and the sums of entries in each row and each column are equal to one respectively (Gagniuc 2017). Denote the set of $n \times n$ doubly stochastic matrices by $\mathcal{D}_n$ which is also known as Birkhoff polytope, i.e.,

$$\mathcal{D}_n = \left\{ X \in \mathcal{R}^{n\times n} : X \geq 0, X\boldsymbol{e} = \boldsymbol{e}, X^\top \boldsymbol{e} = \boldsymbol{e} \right\}$$

where $X \geq 0$ means each entry of X is nonnegative and $\boldsymbol{e}$ is a column vector of ones. In this paper, we focus on the matrix optimization problem

$$\min_X \frac{1}{2}\|X - A\|_F^2, \quad s.t.\ X \in \mathcal{D}_n \tag{1}$$

which approximate the data matrix $A \in \mathcal{R}^{n\times n}$ by a doubly stochastic matrix in the Frobenius norm. The problem (1) belongs to the general class of matrix nearness problem (Higham 1989), i.e., a problem of projecting a given matrix onto the set of matrices that satisfy a certain property.

Doubly stochastic matrices have recently received much attention in several areas such as statistics, machine learning and computer visions; see (Sinkhorn 1964; Zass and Shashua 2006; Yan, Shen, and Wang 2014; Wang, Nie, and Huang 2016; Linderman et al. 2018; Birdal and Simsekli 2019; Dallakyan and Pourahmadi 2021) and references therein. Since stochastic matrix are often used to describe the transitions of a Markov chain, the connection between doubly stochastic matrix and the transition probability has been discussed in (Seneta 2006; Boyd, Diaconis, and Xiao 2004; Boyd et al. 2009). Moreover, data clustering as one of the most fundamental and important topics in machine learning (Hastie et al. 2009), has been extensively studied for a long time. Recently, graph-based clustering methods which reveal the pairwise relationship with an affinity matrix have attracted great attention. These works (Yang, Corander, and Oja 2016; Wang, Nie, and Huang 2016; Wang et al. 2022; Ding et al. 2022) use the doubly stochastic matrix to normalize the affinity matrix and improve the clustering performance. Meanwhile, it is worth noting that in the attention-based deep neural networks, if the attention matrix is doubly stochastic, the model accuracy could achieve a significant improvement in computer vision and natural language processing tasks (Tay et al. 2020; Sander et al. 2022).

Since classical Birkhoff-von Neumann theorem (Birkhoff 1946; Von Neumann 1953) shows that the set $\mathcal{D}_n$ coincides with the convex hull of the $n \times n$ permutation matrices, and furthermore that the vertices of $\mathcal{D}_n$ are precisely the permutation matrices, the polytope $\mathcal{D}_n$ has been frequently introduced as a convex relaxation to the combinatorial problem involving permutations (Fogel et al. 2013; Lim and Wright 2014), such as Bayesian networks (Dallakyan and Pourahmadi 2021), graph matching problem (Zaslavskiy, Bach, and Vert 2009) and quadratic assignment problem (Lawler 1963), etc.

In general, tackling the relaxation problem requires computing the nearest doubly stochastic matrix. Therefore, it is crucial to develop a fast solver for the problem (1), which determines whether the problem involving doubly stochastic matrix can be solved efficiently.

Related Works

It is straightforward to verify that the set of $\mathcal{D}_n$ is convex. This implies that the primal problem (1) is a special case of convex constrained quadratic optimization, which can be solved using standard methods such as interior-point solvers (Nocedal and Wright 2006). However, these methods scale

poorly with the size of the problem. For instance, attempting to use the state-of-the-art interior-point based commercial solver Gurobi (Gurobi Optimization, LLC 2022) to tackle the problem (1) will result in out-of-memory issues even for medium-sized problems. Meanwhile, the problem (1) can be viewed as a standard projection onto the intersection of two convex sets, where one is generated by two equation constraints and the other is the nonnegative orthant. In (Zass and Shashua 2006), the problem (1) with the positive semidefinite input A has been studied based on the alternating projection method (Escalante and Raydan 2011). In this case, each projection sub-problem can be solved efficiently in closed form. Nevertheless, as shown in (Rontsis and Goulart 2020) classical alternating projection algorithm always converges to a feasible point, but does not necessarily converge to the optimal solution of the problem (1).

Another approach to tackle (1) is to solve its dual problem, and then recover the projection using the relationship between the primal and dual. Recently, Hager and Zhang (Hager and Zhang 2016) developed an algorithm for the problem (1) based on the dual active set strategy (Hager 1992). The algorithm implementation called PPROJ was also developed and the experimental results showed that PPROJ is robust, accurate and fast. However, it should be noted that PPROJ is not tailored for the problem (1), and so it cannot take full advantage of the specific structure. As the derived dual of the problem (1) is a convex optimization problem with a smooth objective function, it can be solved by employing the first-order methods. Jiang, Liu, and Wen (2016) proposed a fast dual gradient method using the Barzilai-Borwein (BB) step size (Barzilai and Borwein 1988) to handle the dual problem. Their method is named dualBB which is often able to find a highly accurate solution faster than the commercial solver MOSEK (ApS 2019). Very recently, Li, Sun, and Toh (2020) proposed an elaborated semismooth Newton method to solve the dual problem. Although the numerical results in (Li, Sun, and Toh 2020) show the semismooth Newton method outperformed its counterparts, including PPROJ and Gurobi etc., by a significant margin, it is worth noting that a regularized parameter has to be embedded in the procedure for finding the Newton direction due to the singularity of the Hessian matrix. Meanwhile, Li, Sun, and Toh (2020) did not show the comparison results between their Newton method and the light-weighted dualBB.

Our Method

In our work, instead of rectifying the Hessian to be positive definite by regularizing it with a unit matrix, we propose an efficient quasi-Newton algorithm to deal with the dual problem. The key idea is to use a cheap Hessian approximation which is built at each step based on the partial information of the true Hessian. Even though Newton's method enjoys a fast rate of convergence theoretically, our algorithm takes much lower cost at each iteration and thereby is more efficient in practice. The main contributions of our work are as follows:

- A structured BFGS algorithm is proposed for the matrix nearness problem (1). By embedding curvature information into the diagonal components of the Hessian, our method requires the comparable cost to the first-order gradient methods. This strategy results in a highly efficient solver to the problem (1).
- To alleviate the pressure of computational cost, we introduce a line search procedure based on Newton's method to choose an appropriate step size. In general, we do not need to calculate the objective function value, but use the existing gradient information to obtain the step size.
- Global convergence is established, and the experimental results demonstrate that our structured BFGS algorithm outperforms both dualBB and the semismooth Newton method in terms of running time.

The rest of this paper is organized as follows. We first derive the dual problem of the nearest doubly stochastic matrix problem. Then we present our structured BFGS method and establish its global convergence. Finally, we evaluate the performance of our algorithm and report the numerical results on both synthetic and real-world data.

Problem Reformulation

As the primal problem (1) has n^2 variables and n^2+2n constraints, it is difficult to scale to large-scale problem when tackling the original formulation directly. In this section, we will derive the dual problem which is an unconstrained optimization problem and has significantly fewer variables.

Introducing Lagrange multipliers α and β for the equality constraints in the problem (1), while keeping the nonnegative constraint $X \geq 0$ intact, the Lagrange dual function takes the form as

$$\min_{X\geq 0} \mathcal{L}(X;\alpha,\beta) = \frac{1}{2}\|X-A\|_F^2 + \alpha^\top (X\boldsymbol{e}-\boldsymbol{e}) + \beta^\top \left(X^\top \boldsymbol{e}-\boldsymbol{e}\right)$$

Applying the zero gradient condition with respect to the (i,j)th variable $X_{i,j}$, and combining the KKT conditions with the nonnegative constraint on $X_{i,j}$, we obtain the primal optimal

$$X_{i,j} = \max\left(0, A_{i,j}-\alpha_i-\beta_j\right).$$

Substituting the expression $X_{i,j}$ in the Lagrange function and omitting the constant term, we can write the dual optimization problem as

$$\min_{\alpha,\beta} \mathcal{F}(\alpha,\beta) = \frac{1}{2}\|\mathcal{P}_+(A-\alpha\boldsymbol{e}^\top - \boldsymbol{e}\beta^\top)\|_F^2 + \langle \alpha+\beta, \boldsymbol{e}\rangle \tag{2}$$

where $\mathcal{P}_+(\cdot)$ denote the projection onto the nonnegative orthant.

Since the objective function in the unconstrained optimization problem (2) is convex and smooth, any vector $x = [\alpha^\top \ \beta^\top]^\top$ satisfying the first-order optimality condition

$$\nabla \mathcal{F}(x) = 0 \tag{3}$$

is a global minimum of $\mathcal{F}$. The gradient vector $\nabla\mathcal{F}(x)$ can be assembled from the partial derivatives with respect to the variables α and β according to the following formula

$$\begin{aligned}\frac{\partial\mathcal{F}}{\partial\alpha} &= -\mathcal{P}_+\left(A-\alpha\boldsymbol{e}^\top-\boldsymbol{e}\beta^\top\right)\boldsymbol{e}+\boldsymbol{e},\\ \frac{\partial\mathcal{F}}{\partial\beta} &= -\mathcal{P}_+\left(A-\alpha\boldsymbol{e}^\top-\boldsymbol{e}\beta^\top\right)^\top\boldsymbol{e}+\boldsymbol{e}.\end{aligned} \tag{4}$$

We note that the gradient $\nabla\mathcal{F}(x)$ is Lipschitz continuous, that is, there exists a constant $L>0$ such that

$$\|\nabla\mathcal{F}(x_1)-\nabla\mathcal{F}(x_2)\|\le L\|x_1-x_2\|,$$

for all $x_1, x_2\in\mathcal{R}^{2n}$.

While the dual variable x in (3) is of dimension $2n$, it requires $O(n^2)$ operations to compute the gradient at each iteration as all elements in matrix A should be accessed. Once we obtain the gradient information $\nabla\mathcal{F}(x)$, with a suitable line search strategy, gradient-based methods can be employed to solve the dual problem (2). The iteration is specified in the form

$$x_{k+1}=x_k+\lambda_k d_k, \tag{5}$$

where the positive scalar λ_k is called the step size and d_k is the search direction. For instance, to ensure the convergence of the classical BFGS algorithm, it is necessary to choose a step size λ_k that satisfies the Wolfe conditions, i.e.,

$$\begin{aligned}\mathcal{F}(x_k+\lambda_k d_k)&\le\mathcal{F}(x_k)+c_1\lambda_k\nabla\mathcal{F}_k^\top d_k,\\ \nabla\mathcal{F}(x_k+\lambda_k d_k)^\top d_k&\ge c_2\nabla\mathcal{F}_k^\top d_k,\end{aligned} \tag{6}$$

with $0<c_1<c_2<1$. It can be seen that function and gradient evaluations are required in line search rules (6) for the problem (2), while each evaluation is performed at the cost of $O(n^2)$ operations.

Recently, a nonmontone line search with the BB step size called dualBB is adopted in the dual gradient method (Jiang, Liu, and Wen 2016). DualBB gives rise to efficient numerical results to tackle the problem (2) and performs a line search at a cost of $O(n)$ operations per iteration. However, it inaccurately estimates the Hessian by the step size times the identity matrix, which make the nonmonotonic behaviour of the function value. On the other hand, Newton's method (Li, Sun, and Toh 2020) is arguably elaborate which enjoys fast convergence rate under appropriate conditions. In the problem (2), it happens that the Hessian is positive semidefinite at each iteration. In order to make the Newton's method well-defined, the Hessian must be rectified by adding a regularized parameter, which will inevitably undermine the theoretical convergence rate. Meanwhile, Newton's method is confronted with heavy computational burden per iteration to solve a linear system and find the Newton direction. In the next section we will propose a structured BFGS algorithm which exploits the curvature information based on the diagonal of the Hessian to solve the dual problem (2) at low cost in computation and storage.

Structured BFGS Procedure for the Dual Problem

Our Proposed Algorithm

The descent direction of the classic BFGS algorithm has the form

$$d_k=-H_k\nabla\mathcal{F}_k, \tag{7}$$

where the positive definite matrix H_k contains the inverse Hessian approximation. The fundamental idea of our method is to update H_k with the diagonal component of the true Hessian and the observed information when computing the gradient.

It can be shown that the objective function $\mathcal{F}(\alpha,\beta)$ is not twice differentiable as its gradient $\nabla\mathcal{F}(\alpha,\beta)$ is not differentiable at $\{(\alpha,\beta)\mid A_{i,j}-\alpha_i-\beta_j=0 \text{ for some } i \text{ and } j\}$. As indicated in (Li, Sun, and Toh 2020) that $\nabla\mathcal{F}(\alpha,\beta)$ is semismooth (Mifflin 1977; Sun and Sun 2002) under the mild assumption, and we obtain

$$\partial^2\mathcal{F}(\alpha,\beta)=\begin{pmatrix}\mathrm{diag}(\mathcal{I}\boldsymbol{e}) & \mathcal{I}\\ \mathcal{I}^\top & \mathrm{diag}(\mathcal{I}^\top\boldsymbol{e})\end{pmatrix},$$

where $\partial^2\mathcal{F}(\alpha,\beta)$ denotes the generalized Hessian of $\mathcal{F}$ at (α,β), i.e., the Clarke subdifferential of $\nabla\mathcal{F}$ at (α,β); matrix $\mathcal{I}\in\mathcal{R}^{n\times n}$ is given by

$$\mathcal{I}_{i,j}=\begin{cases}1, & \text{if } A_{i,j}-\alpha_i-\beta_j>0,\\ 0, & \text{otherwise,}\end{cases}$$

for all $i,j=1,\cdots,n$.

It should be noted that the singularity of $\partial^2\mathcal{F}(\alpha,\beta)$ prevents the direct employment of Newton's method (Li, Sun, and Toh 2020). Thus we propose a structured BFGS method to circumvent this issue. We extract the diagonal components from the generalized Hessian $\partial^2\mathcal{F}(\alpha,\beta)$, and define a diagonal matrix Λ as

$$\Lambda=\left[\mathrm{diag}\left(\max\left(1,\left[\boldsymbol{e}^\top\mathcal{I}^\top\ \boldsymbol{e}^\top\mathcal{I}\right]^\top\right)\right)\right]^{-1}, \tag{8}$$

where the max is applied in the component-wise sense and we set the diagonal entries greater than zero to avoid the singularity. From the definition (8) we can immediately obtain the following property about the positive matrix Λ.

Proposition 1 *Let the positive diagonal matrix Λ as given by (8). Then each diagonal element*

$$\frac{1}{n}\le\Lambda_{ii}\le 1,\ \ i=1,\cdots,n.$$

Now we are ready to derive our scheme. As BFGS method is the most popular quasi-Newton algorithm (Nocedal and Wright 2006), we follow the BFGS procedure to build curvature information into the diagonal matrix Λ so that it approaches the inverse Hessian. To determine the updated approximation H_{k+1} at iteration k, we first define

$$s_k=x_{k+1}-x_k,\ y_k=\nabla\mathcal{F}_{k+1}-\nabla\mathcal{F}_k,$$

and then solve the following problem

$$\begin{aligned}&\min_H\ \|H-\Lambda_k\|\\ &s.t.\ H=H^\top,\quad Hy_k=s_k.\end{aligned} \tag{9}$$

The unique solution H_{k+1} to (9) is given by

$$H_{k+1} = \left(I - \rho_k s_k y_k^\top\right) \Lambda_k \left(I - \rho_k y_k s_k^\top\right) + \rho_k s_k s_k^\top, \quad (10)$$

where $\rho_k = 1/(s_k^\top y_k)$ and I denotes the identity matrix. The identity matrix is the square diagonal matrix with ones on the main diagonal and zeros elsewhere. The following proposition reveals a good property of the updated Hessian approximation H_{k+1}.

Proposition 2 *Suppose that Λ_k is positive definite. Then the update formula (10) is well defined and H_{k+1} is positive definite if $y_k^\top s_k > 0$.*

Proof: First, we have $\rho_k > 0$ if $y_k^\top s_k > 0$. Thus, (10) is well defined. Given any nonzero vector z, we obtain that

$$z^\top H_{k+1} z = \hat{z}^\top \Lambda_k \hat{z} + \rho_k (z^\top s_k)^2,$$

where $\hat{z} = z - \rho_k y_k s_k^\top z$.

In the case of $z^\top s_k = 0$, we have $\hat{z} = z$ and $z^\top H_{k+1} z \geq \|z\|^2/n > 0$. Otherwise, the conclusion holds immediately as two positive terms on the right-hand size. ■

Note that $y_k^\top s_k > 0$ is called the *curvature condition* in standard BFGS algorithm, and it is guaranteed if the step size λ_k satisfies the Wolfe conditions (6).

The above proposition shows that H_{k+1} inherits the positive definiteness of Λ_k. Thus, d_k in (7) is a descent direction at every iteration which forms the basis of our method and guarantees the objective function in (2) can be reduced along this direction. Moreover, in order to ensure the convergence of our algorithm, it is necessary to verify the angle θ_k between the descent direction d_k and the negative gradient $-\nabla\mathcal{F}_k$ stays uniformly bounded away from 90°, that is,

$$\cos\theta_k = \frac{-\nabla\mathcal{F}_k^\top d_k}{\|\nabla\mathcal{F}_k\|\|d_k\|} \geq 1/n. \quad (11)$$

Our procedure is specified as Algorithm 1. The standard BFGS algorithm and our method both combine two successive iterates x_k, x_{k+1} together with the corresponding gradients $\nabla\mathcal{F}_k, \nabla\mathcal{F}_{k+1}$, but they yield curvature information in different ways. While the former uses the existing knowledge to update the current Hessian approximation, our method embeds the most recently observed information into the diagonal matrix Λ_k of the true Hessian. It should be noted that the approximation of the Hessian is usually dense in the standard BFGS procedure, even when the true Hessian is sparse. Thus, the BFGS algorithm requires at least $O(n^2)$ arithmetic operations and $O(n^2)$ storage demands per iteration. By contrast, our method allows approximate steps to be calculated at lower cost in computation and storage as Λ_k in (10) is a diagonal matrix. Since our method does not require explicit knowledge of the inverse Hessian approximation, it requires only $O(n)$ operations to compute the search direction d_k and update the matrix H_{k+1}, which is significantly fewer than the cost $O(n^2)$ to evaluate $\mathcal{F}$ and $\nabla\mathcal{F}$.

Newton-based Line Search

A simple and popular strategy, i.e., backtracking line search, is usually employed to find the step size. It begins with an initial guess of step size, e.g., $\lambda = 1.0$, and decreases the value repeatedly until some user-specified condition is satisfied. In this procedure, additional evaluations of the objective function $\mathcal{F}$ are required and each evaluation takes $O(n^2)$ operations. In this subsection, we describe a Newton-based line search for finding an appropriate step size to satisfy the Wolfe conditions (6).

Algorithm 1: Structured BFGS Algorithm for the Dual Problem (2)

Require: Given starting point $x_0 = [\alpha_0^\top \ \beta_0^\top]^\top$, set convergence tolerance $\epsilon > 0$, and $k = 0$
1: Compute the gradient $\nabla\mathcal{F}(x_0)$ and inverse diagonal Hessian information Λ_0 of $\mathcal{D}(\alpha_0, \beta_0)$
2: Set the inverse Hessian approximation $H_0 = \Lambda_0$
3: **while** $\|\nabla\mathcal{F}_k\| > \epsilon$ **do**
4: Compute search direction $d_k = -H_k\nabla\mathcal{F}_k$
5: **if** $\cos\theta_k < 1/n$ **then**
6: $d_k = -\Lambda_k\mathcal{F}_k$
7: **end if**
8: Choose the step size λ_k to satisfy the Wolfe conditions (6) from Algorithm 2
9: Update $x_{k+1} = x_k + \lambda_k d_k$
10: Compute the gradient $\nabla\mathcal{F}_{k+1}$and the diagonal information Λ_{k+1} in the true Hessian
11: Set $s_k = x_{k+1} - x_k, y_k = \nabla\mathcal{F}_{k+1} - \nabla\mathcal{F}_k$
12: Compute the inverse Hessian approximation H_{k+1} by means of (10)
13: Set $k = k + 1$
14: **end while**
Ensure: x_k

For updating α_k to α_{k+1} and β_k to β_{k+1} in the problem (2), we need to solve the following univariate sub-problem

$$\begin{aligned}&\min_{\lambda>0} \mathcal{D}(\lambda) := \langle \alpha + \lambda d^\alpha + \beta + \lambda d^\beta, \boldsymbol{e}\rangle + \\ &\frac{1}{2}\left\|\mathcal{P}_+\left(A - \alpha\boldsymbol{e}^\top - \boldsymbol{e}\beta^\top - \lambda d^\alpha \boldsymbol{e}^\top - \lambda \boldsymbol{e} d^{\beta\top}\right)\right\|_F^2,\end{aligned} \quad (12)$$

where d^α denotes the first half of search direction d, and d^β denotes the second half of d; we drop the subscript k from the quantities α_k, β_k and d_k for simplicity.

Although $D(\lambda)$ is not twice differentiable, it is a semismooth funtion. We can compute its first derivative and the generalized second derivative as follows

$$\begin{aligned}\mathcal{D}'(\lambda) &= \langle d^\alpha + d^\beta, \boldsymbol{e}\rangle + \\ &\sum_{(i,j)\in\mathcal{I}(\lambda)} \left(A_{i,j} - \alpha_i - \beta_j - \lambda d_i^\alpha - \lambda d_j^\beta\right)\left(-d_i^\alpha - d_j^\beta\right), \\ \mathcal{D}''(\lambda) &= \sum_{(i,j)\in\mathcal{I}(\lambda)} \left(d_i^\alpha + d_j^\beta\right)^2,\end{aligned}$$

where

$$\begin{aligned}\mathcal{I}(\lambda) = \{(i,j) :& A_{i,j} - \alpha_i - \beta_j - \lambda d_i^\alpha - \lambda d_j^\beta > 0, \\ & i, j = 1, \cdots, n\}.\end{aligned}$$

The Newton method to solve the problem (12) updates λ

Algorithm 2: Newton-based Line Search Method for Solving the Sub-problem (12)

Require: Given the search direction d, and set the initial step size $\lambda = 0$

```
while True do
    Compute the derivatives D'(λ) and D''(λ)
    Update the step size λ via (13)
    if λ satisfies the Wolfe conditions (6) then
        break;
    end if
end while
```

Ensure: λ

by the following way

$$\lambda \leftarrow \lambda - \frac{\mathcal{D}'(\lambda)}{\mathcal{D}''(\lambda)} \tag{13}$$

until λ satisfies the Wolfe conditions (6). Algorithm 2 lists the details of our line search procedure. Using the fact that $\|\nabla\mathcal{F}\|$ is greater than zero before Algorithm 1 terminates, we can conclude that $d \neq 0$ and $\mathcal{D}''(\lambda) > 0$, which means our Newton iteration (13) is well-defined.

While our line search procedure is an exact method that can terminate in finite steps, solving the sub-problem (12) exactly is too expensive and we need just an inexact solution. Meanwhile, with the initialization of z to zero in (13), the line search procedure in practice only takes one iteration to satisfy the conditions. In particular,

$$\lambda \leftarrow -\frac{\mathcal{D}'(0)}{\mathcal{D}''(0)} = -\frac{\nabla\mathcal{F}(x_k)^\top d_k}{\mathcal{D}''(0)}.$$

One advantage in this situation is that there is no need to evaluate objective function value and gradient, we can use the existing knowledge to get the step size which takes $O(2n + nnz(\mathcal{I}))$ operations.

Convergence Analysis

In this section we present some convergence results of our structured BFGS algorithm which is proposed in Algorithm 1. It is well known that the convergence of an optimization algorithm cannot hold independently of the choice of d_k. We first show that our search directions are never too close to orthogonality with the gradient.

Proposition 3 *Let θ_k as the angle between the descent direction d_k and the negative gradient $\nabla\mathcal{F}(x_k)$. Then for all the steps k we have $\cos\theta_k \geq 1/n > 0$.*

Proof: When it occurs that $\cos\theta_k < 1/n$ in line 5 of Algorithm 1, the search direction is replaced by $d_k = -\Lambda_k\nabla\mathcal{F}_k$. Thus we obtain

$$\cos\theta_k = \frac{\nabla\mathcal{F}_k^\top\Lambda_k\nabla\mathcal{F}_k}{\|\nabla\mathcal{F}_k\|\|\Lambda_k\nabla\mathcal{F}_k\|} \geq \frac{1}{n}.$$

The last inequality comes from Proposition 1. ■

The above proposition tells that $\cos\theta_k$ is sufficiently positive which means d_k is a definite descent direction. Before establishing the global convergence of our Algorithm 1, we introduce the following Zoutendijk's result.

Lemma 4 (Theorem 3.2, (Nocedal and Wright 2006)) *Suppose that $\mathcal{F}$ is bounded below and that $\mathcal{F}$ is continuously differentiable. Assume also the gradient $\nabla\mathcal{F}$ is Lipschitz continuous and consider any iteration of the form (5), where d_k is a descent direction and λ_k satisfies the Wolfe conditions (6). Then*

$$\sum_{k\geq 0}\cos^2\theta_k\|\nabla\mathcal{F}_k\|^2 < \infty. \tag{14}$$

From Proposition 3 and Lemma 4, the following Theorem 5 which shows the global convergence of Algorithm 1 can be immediately obtained.

Theorem 5 *Let x_0 be a starting point. Then the sequence $\{x_k\}$ generated by Algorithm 1 converges to the minimizer x^* of $\mathcal{F}(x)$, i.e., $\nabla\mathcal{F}(x^*) = 0$.*

Experimental Results

In this section, we present numerical results of our proposed structural BFGS (s-BFGS) algorithm for the matrix optimization problem (1). In particular, we make the running time comparison against two state-of-the-art methods: the dual gradient method called dualBB and the semismooth Newton algorithm called SSNCG1. Since experimental results in (Jiang, Liu, and Wen 2016; Li, Sun, and Toh 2020) have shown that dualBB and SSNCG1 outperformed all other methods in terms of efficiency, the goal of our work is to prove our superiority over the two methods.

For the fairness of experimental comparison, we terminate three algorithms in the first two experiments with the same stopping tolerance, i.e., $\|\nabla\mathcal{F}_k\| \leq \epsilon = 10^{-12}$. All the algorithms have been implemented in MATLAB R2019b and run on a 3.00-GHz Intel Core i9 Linux machine with 128GB memory. Our code will be released publicly on the github[1] for reproducing all the results of this section.

Numerical Results on the Synthetic Data

We investigate first the scalability of our Algorithm 1 on synthetic data matrix. Namely, we randomly generate a data matrix $A \in \mathbb{R}^{n\times n}$ while varying $n \in \{10^3, 5\times10^3, 10^4, 1.5\times10^4, 2\times10^4, 2.5\times10^4\}$. All the data entries are sampled from the standard normal distribution using the MATLAB command: $A = \text{randn}(n)$.

The iteration numbers and computation times for the three algorithms to reach the same stopping criterion are demonstrated in Figure 1. Since the dualBB is a non-monotonic descent algorithm, the iteration numbers it requires is non-monotonically increasing as the data size increases in Figure 1. We can observe that the time cost of all three methods increases with the number of dimensions n. Although our s-BFGS algorithm requires more iterations than Newton's method, it takes significantly less time due to its low computational cost per iteration.

Figure 2 plots the gradient norm against the iteration counts and computation time of three candidates. It can be seen that Newton's method exhibits super-linear convergence behaviour when approaching the optimal solution,

[1] https://github.com/djchu/sbfgs4dsm

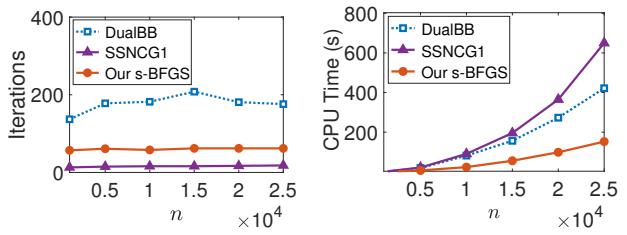

Figure 1: Scalability comparison among the dual gradient-type method dualBB, the semismooth Newton algorithm SSNCG1 and our proposed s-BFGS method. Left: number of iterations until convergence by dimension n. Right: running time by dimension n. Time is in seconds.

and it takes the least iterations among the three methods. However, due to the low computational cost at each iteration, our s-BFGS algorithm spends the least time than two other counterparts. Meanwhile, since our s-BFGS method exploits the second-order information, it has fewer iterations than dualBB.

Numerical Results on the Real Data Sets

In this subsection, we test 6 instances of the given matrices A which are derived from the real LIBSVM data sets at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Similarly as in (Li, Sun, and Toh 2020), we normalize each data point to have a unit l_2-norm and use the following RBF kernel to generate A, i.e.,

$$A_{i,j} = \exp\left(\frac{-\|\mathrm{x}_i - \mathrm{x}_j\|^2}{\sigma^2}\right), \quad \forall 1 \le i, j \le n, \qquad (15)$$

where we use the notation x_i to denote the ith observation in the data set and this should not cause any confusion with the variable x in our Algorithm 1. We set $\sigma = 1.0$ in this subsection.

Table 1 reports the numerical results obtained by dualBB, Newton method and our s-BFGS. It can be seen that the experimental results are similar to the results on the synthetic data. The Newton method requires the least number of iterations, but it is at a disadvantage in the comparison of the computation time. Meanwhile, our s-BFGS algorithm using the second-order information not only has fewer iterations than dualBB, but also spends the least time.

Data Set	n	dualBB	SSNCG1	Our s-BFGS
gisette	6000	23.01(115)	24.85(**13**)	**5.10**(32)
mushrooms	8124	52.24(117)	44.09(**13**)	**12.29**(45)
a6a	11220	32.64(71)	98.57(**15**)	**18.99**(37)
a7a	16100	49.31(78)	200.34(**15**)	**39.65**(38)
rcv1.binary	20242	207.31(162)	846.32(**22**)	**87.86**(54)
a8a	22696	310.16(108)	388.41(**15**)	**79.77**(39)

Table 1: Running time (s) comparison on real data sets. The integers in parentheses indicate the number of iterations.

Application to Spectral Clustering Problems

The goal of spectral clustering is to arrange unlabeled data to clusters, where similar data points hopefully get assigned to the same cluster. Here, the affinity matrix C consists of each element $C_{i,j}$ representing the degree of pairwise similarity between samples x_i and x_j. In general, there are three crucial steps that affect the performance of a spectral algorithm: (i) the construction of the affinity matrix, (ii) the normalization of the affinity matrix, and (iii) the particular clustering algorithm.

Recently, Zass and Shashua (2006) suggested a Frobenius norm normalization scheme which can be formulated as the following optimization problem,

$$\min_X \frac{1}{2}\|X - A\|_F^2, \ s.t.\ X \ge 0, X\boldsymbol{e} = \boldsymbol{e}, X = X^\top. \quad (16)$$

While normalization (16) leads to superior clustering performance over various standardized tests, it is indeed a special case of the generic matrix nearness problem (1), as the latter does not require the input matrix A is symmetric. It is not difficult to verify that when A is symmetric, the optimal X to the problem (1) is also symmetric, and X is also the solution of the problem (16). In this subsection, we show that our s-BFGS algorithm can efficiently solve the normalization problem (16).

In order to solve the problem (16), Zass and Shashua defined two sub-problems. The first considers equality constraints,

$$\min_X \frac{1}{2}\|X - A\|_F^2, \ s.t.\ X\boldsymbol{e} = \boldsymbol{e}, X = X^\top,$$

and the second deals with bound constraints,

$$\min_X \frac{1}{2}\|X - A\|_F^2, \ s.t.\ X \ge 0.$$

Since both of the above two sub-problems enjoy closed-form solution, an alternating projection procedure is developed and takes the following iterative form,

$$\hat{X}_k = X_k + \frac{1}{n}\left(\boldsymbol{e}\boldsymbol{e}^\top - X_k\boldsymbol{e}\boldsymbol{e}^\top - \boldsymbol{e}\boldsymbol{e}^\top X_k\right) + \frac{\boldsymbol{e}^\top X_k \boldsymbol{e}}{n^2}\boldsymbol{e}\boldsymbol{e}^\top,$$
$$X_{k+1} = \mathcal{P}_+(\hat{X}_k). \qquad (17)$$

We should note that although the simple alternating projections (17) is only guaranteed to converge to a feasible point for the problem (16), it is still a popular method for solving the graph-based clustering tasks (Wang, Nie, and Huang 2016; Wang et al. 2022). Since better clustering quality has been presented in (Zass and Shashua 2006), we just compare the efficiency between our method and the alternating projection algorithm.

We follow the setting of alternating projection algorithm (Zass and Shashua 2006) on six real data sets listed in Table 1. The RBF kernel (15) is used to create the affinity matrix for all the data sets. To show the merits of our s-BFGS algorithm, we consider the CPU time used in the relative parameter difference between parameter vectors in consecutive iterations, that is,

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_{k+1}\|_F} \le 10^{-4}.$$

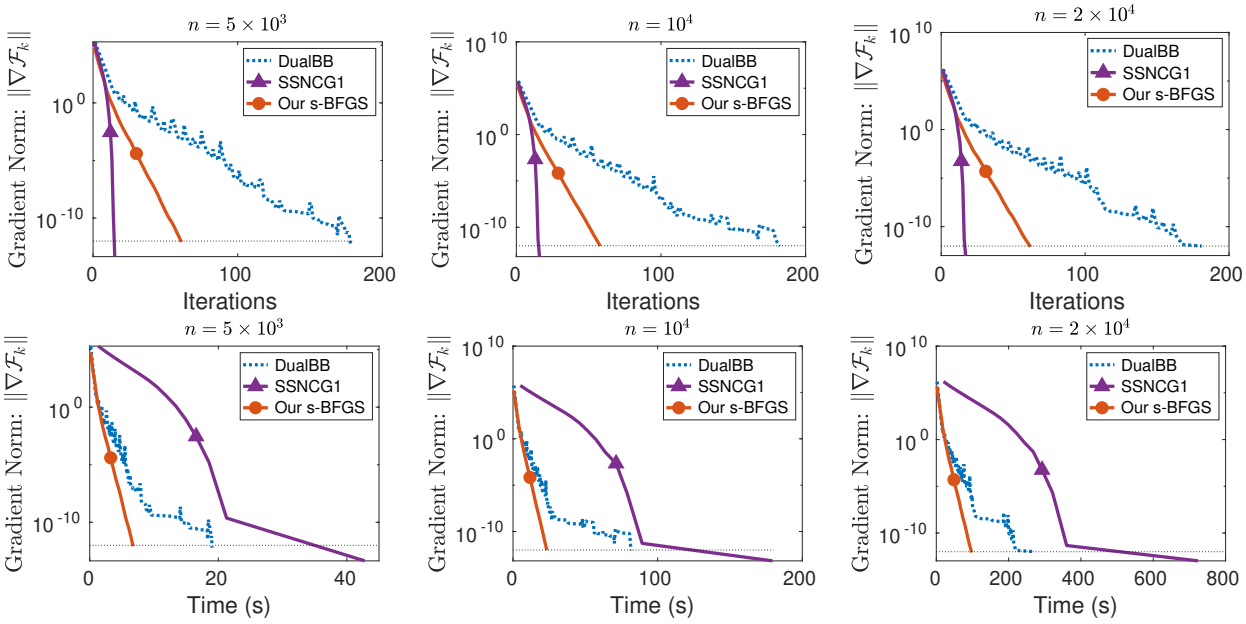

Figure 2: Typical convergence behaviours of dualBB, SSNCG1 and our s-BFGS. Each of the columns contains two plots with the same dimension n. The top row presents the gradient norm $\|\nabla \mathcal{F}_k\|$ as a function of iteration counts k. The bottom row focuses on the gradient norm vs. running time. The dotted line indicates the stopping tolerance ϵ. Time is in seconds.

Data Set	$\sigma = 2.0$			$\sigma = 4.0$			$\sigma = 6.0$		
	Alt. Proj.	s-BFGS	Speedup	Alt. Proj.	s-BFGS	Speedup	Alt. Proj.	s-BFGS	Speedup
gisette	27.88	1.74	16.02	17.65	1.45	12.17	11.99	1.29	9.29
mushrooms	145.87	3.96	36.84	110.77	3.41	32.48	84.97	3.18	26.92
a6a	314.58	7.32	42.97	251.47	6.48	38.81	197.41	5.84	33.80
a7a	950.22	14.90	63.77	795.58	12.91	61.63	640.41	11.90	53.82
rcv1.binary	763.79	27.62	27.65	741.20	19.65	37.72	644.92	16.40	39.32
a8a	2713.31	30.61	88.64	2391.95	26.68	89.65	1963.08	24.83	79.06

Table 2: Running times (s) of solving the affinity matrix normalization problem (16) on real data sets.

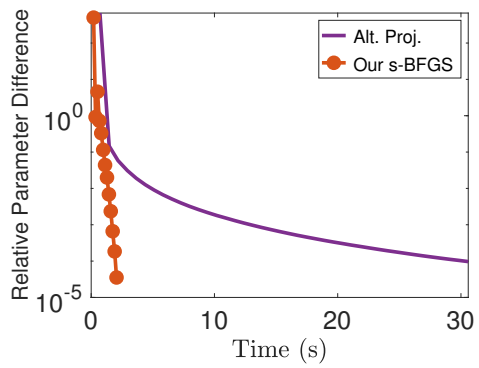

Figure 3: Convergence comparison of alternating projections and our s-BFGS on data set gisette.

Figure 3 shows that our s-BFGS algorithm enjoys much faster convergence rate than the alternative projection algorithm on the data set gisette with the kernel parameter $\sigma = 1.0$.

Since the kernel parameters are typically tuned to achieve the best clustering performance, we also report the acceleration performance of our s-BFGS method with various σ for RBF kernel as illustrated in Table 2.

Conclusions

In this paper we propose an efficient structured BFGS algorithm for finding the optimal approximation to the input data matrix over the Birkhoff polytope. By incorporating the curvature information into the diagonal components of the true Hessian, our algorithm takes only little additional cost to produce the descent direction. We also develop a Newton-based line search method to choose an appropriate step size to satisfy the Wolfe conditions. Empirical results on both artificial data and real-world data sets show that our method outperforms the state-of-the-art solvers. The application of our algorithm to spectral clustering problem for normalizing the affinity matrix illustrates the great potential in doubly stochastic matrix related tasks.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. Dejun Chu would like to acknowledge the funding by Anhui Provincial Natural Science Foundation (No. 1908085MF193). Shiliang Sun is supported by the National Natural Science Foundation of China (NSFC) under Project 62076096 and Shanghai Municipal Project 20511100900. Qing Tao is supported by the NSFC Project 62076252.

References

ApS, M. 2019. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*

Barzilai, J.; and Borwein, J. M. 1988. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1): 141–148.

Birdal, T.; and Simsekli, U. 2019. Probabilistic permutation synchronization using the Riemannian structure of the Birkhoff polytope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11105–11116.

Birkhoff, G. 1946. Three observations on linear algebra. *Univ. Nac. Tacuman, Rev. Ser. A*, 5: 147–151.

Boyd, S.; Diaconis, P.; Parrilo, P.; and Xiao, L. 2009. Fastest mixing Markov chain on graphs with symmetries. *SIAM Journal on Optimization*, 20(2): 792–819.

Boyd, S.; Diaconis, P.; and Xiao, L. 2004. Fastest mixing Markov chain on a graph. *SIAM review*, 46(4): 667–689.

Dallakyan, A.; and Pourahmadi, M. 2021. Learning Bayesian Networks through Birkhoff Polytope: A Relaxation Method. *arXiv preprint arXiv:2107.01658*.

Ding, T.; Lim, D.; Vidal, R.; and Haeffele, B. D. 2022. Understanding Doubly Stochastic Clustering. In *International Conference on Machine Learning*, 5153–5165. PMLR.

Escalante, R.; and Raydan, M. 2011. *Alternating projection methods*. SIAM.

Fogel, F.; Jenatton, R.; Bach, F.; and d'Aspremont, A. 2013. Convex Relaxations for Permutation Problems. In Burges, C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Gagniuc, P. A. 2017. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.

Hager, W. W. 1992. The dual active set algorithm. *Advances in optimization and parallel computing*, 137–142.

Hager, W. W.; and Zhang, H. 2016. Projection onto a polyhedron that exploits sparsity. *SIAM Journal on Optimization*, 26(3): 1773–1798.

Hastie, T.; Tibshirani, R.; Friedman, J. H.; and Friedman, J. H. 2009. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Higham, N. J. 1989. Matrix Nearness Problems and Applications. In Gover, M. J. C.; and Barnett, S., eds., *Applications of Matrix Theory*, 1–27. Oxford University Press.

Jiang, B.; Liu, Y.-F.; and Wen, Z. 2016. Lp-norm regularization algorithms for optimization over permutation matrices. *SIAM Journal on Optimization*, 26(4): 2284–2313.

Lawler, E. L. 1963. The quadratic assignment problem. *Management science*, 9(4): 586–599.

Li, X.; Sun, D.; and Toh, K.-C. 2020. On the efficient computation of a generalized Jacobian of the projector over the Birkhoff polytope. *Mathematical Programming*, 179(1): 419–446.

Lim, C. H.; and Wright, S. 2014. Beyond the Birkhoff polytope: Convex relaxations for vector permutation problems. *Advances in neural information processing systems*, 27.

Linderman, S.; Mena, G.; Cooper, H.; Paninski, L.; and Cunningham, J. 2018. Reparameterizing the Birkhoff polytope for variational permutation inference. In *International Conference on Artificial Intelligence and Statistics*, 1618–1627. PMLR.

Mifflin, R. 1977. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6): 959–972.

Nocedal, J.; and Wright, S. J. 2006. *Numerical Optimization*. New York, NY, USA: Springer, 2e edition.

Rontsis, N.; and Goulart, P. 2020. Optimal approximation of doubly stochastic matrices. In *International Conference on Artificial Intelligence and Statistics*, 3589–3598. PMLR.

Sander, M. E.; Ablin, P.; Blondel, M.; and Peyré, G. 2022. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, 3515–3530. PMLR.

Seneta, E. 2006. *Non-negative matrices and Markov chains*. New York: Springer.

Sinkhorn, R. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2): 876–879.

Sun, D.; and Sun, J. 2002. Semismooth matrix-valued functions. *Mathematics of Operations Research*, 27(1): 150–169.

Tay, Y.; Bahri, D.; Yang, L.; Metzler, D.; and Juan, D.-C. 2020. Sparse Sinkhorn attention. In *International Conference on Machine Learning*, 9438–9447. PMLR.

Von Neumann, J. 1953. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2(0): 5–12.

Wang, Q.; He, X.; Jiang, X.; and Li, X. 2022. Robust bistochastic graph regularized matrix factorization for data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1): 390–403.

Wang, X.; Nie, F.; and Huang, H. 2016. Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix. In *Proceedings of the 22nd ACM SIGKDD International conference on Knowledge discovery and data mining*, 1245–1254.

Yan, Y.; Shen, C.; and Wang, H. 2014. Efficient semidefinite spectral clustering via Lagrange duality. *IEEE Transactions on image processing*, 23(8): 3522–3534.

Yang, Z.; Corander, J.; and Oja, E. 2016. Low-rank doubly stochastic matrix decomposition for cluster analysis. *The Journal of Machine Learning Research*, 17(1): 6454–6478.

Zaslavskiy, M.; Bach, F.; and Vert, J.-P. 2009. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12): 2227–2242.

Zass, R.; and Shashua, A. 2006. Doubly stochastic normalization for spectral clustering. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 1569–1576.