Learning a Bi-Stochastic Data Similarity Matrix

Fei Wang
Department of Statistical Science
Cornell University
Ithaca, NY 14853, USA
fw83@cornell.edu

Ping Li
Department of Statistical Science
Cornell University
Ithaca, NY 14853, USA
pingli@cornell.edu

Arnd Christian König
Microsoft Research
Microsoft Coopration
Redmond, WA 98052, USA
chrisko@microsoft.com

***Abstract*—An idealized clustering algorithm seeks to learn a cluster-adjacency matrix such that, if two data points belong to the same cluster, the corresponding entry would be 1; otherwise the entry would be 0. This integer (1/0) constraint makes it difficult to find the optimal solution. We propose a relaxation on the cluster-adjacency matrix, by deriving a bi-stochastic matrix from a data similarity (e.g., kernel) matrix according to the Bregman divergence. Our general method is named the *Bregmanian Bi-Stochastication* (BBS) algorithm.**

We focus on two popular choices of the Bregman divergence: the Euclidian distance and the KL divergence. Interestingly, the BBS algorithm using the KL divergence is equivalent to the Sinkhorn-Knopp (SK) algorithm for deriving bi-stochastic matrices. We show that the BBS algorithm using the Euclidian distance is closely related to the relaxed k-means clustering and can often produce noticeably superior clustering results than the SK algorithm (and other algorithms such as Normalized Cut), through extensive experiments on public data sets.

I. Introduction

Clustering [13], [6], which aims to organize data in an unsupervised fashion, is one of the fundamental problems in data mining and machine learning. The basic goal is to group the data points into clusters such that the data in the same cluster are "similar" to each other while the data in different clusters are "different" from each other.

In this paper, we view clustering from the perspective of matrix approximation. Suppose we are given a data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, which comes from k clusters. We can denote the cluster memberships by an $n \times k$ matrix $\mathbf{F}$, such that

$$F_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in \pi_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where π_j denotes the j-th cluster. It is often more convenient to proceed with the scaled version $\widetilde{\mathbf{F}}$ [17][24], such that

$$\widetilde{F}_{ij} = \begin{cases} 1/\sqrt{n_j}, & \text{if } \mathbf{x}_i \in \pi_j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $n_j = |\pi_j|$ is the cardinality of cluster π_j. Note that $\widetilde{\mathbf{F}}$ has (at least) the following properties (constraints):

$$\tilde{\mathbf{F}} \geqslant 0 \text{ (i.e., } \widetilde{F}_{ij} \geqslant 0 \ \forall\ i,j), \quad \widetilde{\mathbf{F}}^\top \widetilde{\mathbf{F}} = \mathbf{I}, \quad \left(\widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top\right)\mathbf{1} = \mathbf{1},$$

where $\mathbf{1} \in \mathbb{R}^{n\times 1}$ is an all-one vector, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix.

If we define $\mathbf{G} = \widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top$, we can hope to discover the cluster structure of $\mathcal{X}$ from $\mathbf{G}$. The constraints on $\widetilde{\mathbf{F}}$ can be transferred to the constraints on $\mathbf{G}$ as

$$\mathbf{G} \geqslant 0, \quad \mathbf{G} = \mathbf{G}^\top, \quad \mathbf{G}\mathbf{1} = \mathbf{1} \tag{3}$$

In other words, $\mathbf{G}$ is a symmetric, nonnegative, and *bi-stochastic* (also called *doubly stochastic*) matrix [12].

A. Deriving a Bi-Stochastic Matrix from a Similarity Matrix

The bi-stochastic matrix $\mathbf{G}$, constructed from the cluster-membership matrix ($\mathbf{F}$ or $\tilde{\mathbf{F}}$) can be viewed as a special type of *similarity matrix*. Naturally, one might conjecture: *If we relax the integer (0/1) constraint on $\mathbf{F}$, can we still derive a (useful) bi-stochastic matrix from a similarity matrix?*

For example, a popular family of data similarity matrix is the *Gaussian kernel* matrix, $\mathbf{K} \in \mathbb{R}^{n\times n}$, where each entry

$$K_{ij} = \exp\left(-\frac{1}{\gamma}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \quad \gamma > 0 \tag{4}$$

Here, γ is a tuning parameter. Obviously, an arbitrary similarity matrix can not be guaranteed to be bi-stochastic. For a given similarity matrix, there are multiple ways to derive a bi-stochastic matrix. We first review a straightforward solution known as the *Sinkhorn-Knopp (SK)* algorithm.

B. The Sinkhorn-Knopp (SK) Algorithm

The following *Sinkhorn-Knopp Theorem* [18] says that, under mild regularity conditions, one can construct a bi-stochastic matrix from a similarity matrix.

Theorem (Sinkhorn-Knopp) *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a nonnegative square matrix. A necessary and sufficient condition that there exists a bi-stochastic matrix $\mathbf{P}$ of the form: $\mathbf{P} = \mathbf{U}\mathbf{A}\mathbf{V}$, where $\mathbf{U}$ and $\mathbf{V}$ are diagonal matrices with positive main diagonals, is that $\mathbf{A}$ has total support. If $\mathbf{P}$ exists, then it is unique. $\mathbf{U}$ and $\mathbf{V}$ are also unique up to a scalar multiple if and only if $\mathbf{A}$ is fully indecomposable.*

Based on this theorem, [18] proposed an method called the *Sinkhorn-Knopp* (SK) algorithm to obtain a bi-stochastic matrix from a nonnegative matrix $\mathbf{A}$, by generating a sequence of matrices whose columns and rows are normalized alternatively. The limiting matrix is bi-stochastic. In particular,

if $\mathbf{A}$ is symmetric, then the resulting matrix $\mathbf{P} = \mathbf{UAV}$ is also symmetric with $\mathbf{U}$ and $\mathbf{V}$ being equal (up to a constant multiplier). The following example illustrates the procedure:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.8 & 0.6 \\ 0.8 & 1 & 0.4 \\ 0.6 & 0.4 & 1 \end{bmatrix}$$
$$\longrightarrow \begin{bmatrix} 0.4167 & 0.3636 & 0.3000 \\ 0.3333 & 0.4545 & 0.2000 \\ 0.2500 & 0.1818 & 0.5000 \end{bmatrix} \longrightarrow \begin{bmatrix} 0.3857 & 0.3366 & 0.2777 \\ 0.3374 & 0.4601 & 0.2025 \\ 0.2683 & 0.1951 & 0.5366 \end{bmatrix}$$
$$\longrightarrow ...$$
$$\longrightarrow \begin{bmatrix} 0.3886 & 0.3392 & 0.2722 \\ 0.3392 & 0.4627 & 0.1980 \\ 0.2722 & 0.1980 & 0.5297 \end{bmatrix} = \mathbf{P}$$

The SK algorithm is not the unique construction. In statistics, this procedure is also known as the *iterative proportional scaling* algorithm [5], [20].

C. Connection to the Normalized Cut (Ncut) Algorithm

Interestingly, the well-known *Normalized Cut (Ncut)* algorithm [17] can be viewed as a one-step construction towards producing bi-stochastic matrices. The Ncut algorithm normalizes a similarity matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ with $\mathbf{D} = \text{diag}(\mathbf{K1})$, where $\mathbf{1} \in \mathbb{R}^{n\times 1}$ is an all-one vector, to be

$$\widetilde{\mathbf{K}} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2} \tag{5}$$

[23] showed that if one keeps normalizing $\mathbf{K}$ with

$$\mathbf{K}^{(t+1)} = (\mathbf{D}^{(t)})^{-1/2}\mathbf{K}^{(t)}(\mathbf{D}^{(t)})^{-1/2}, \tag{6}$$

then $\mathbf{K}^{(\infty)}$ will be bi-stochastic.

D. Our Proposed General Framework: BBS

In this paper, we propose to obtain a bi-stochastic matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ from some initial similarity matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, by solving the following optimization problem

$$\begin{aligned} \min_{\mathbf{G}} \quad & D_\phi(\mathbf{G},\mathbf{K}) = \textstyle\sum_{ij} D_\phi(G_{ij},K_{ij}) \\ s.t. \quad & \mathbf{G} \geqslant 0, \ \ \mathbf{G} = \mathbf{G}^\top, \ \ \mathbf{G1} = \mathbf{1} \end{aligned} \tag{7}$$

where

$$D_\phi(x,y) \triangleq \phi(x) - \phi(y) - \nabla\phi(y)(x-y), \tag{8}$$

is the *Bregman divergence* between x and y with ϕ being a strictly convex function. The problem (7) is a standard convex optimization program. We name the solution $\mathbf{G}$ the *Bregmanian Bi-Stochastication* (*BBS*) of $\mathbf{K}$.[1]

Two choices of the Bregman divergence D_ϕ are popular:

1) $\phi(x) = x^2/2$: (squared) Euclidian distance,

2) $\phi(x) = x\log x - x$: Kullback-Leibler (KL) divergence.

It can be shown that the SK algorithm is equivalent to BBS using KL divergence. We will demonstrate that BBS with $\phi(x) = x^2/2$ often produces superior clustering results over the SK algorithm (and other algorithms such as Ncut).

[1]Also see the work on matrix nearness in Bregman divergence without the bi-stochastic constraint [7].

II. Bregmanian Bi-Stochastication (BBS)

The BBS algorithm seeks a bi-stochastic matrix $\mathbf{G}$ which optimally approximates $\mathbf{K}$ in the Bregman divergence sense, by solving the optimization problem (7). For the two popular choices of the Bregman divergence D_ϕ in Eq. (8), we study specially designed optimization strategies, for better insights.

A. $\phi(x) = x^2/2$

For this choice of $\phi(x)$, we have

$$\begin{aligned} & D_\phi(\mathbf{G},\mathbf{K}) = \textstyle\sum_{ij} D_\phi(G_{ij},K_{ij}) \\ & = \textstyle\sum_{ij} \frac{1}{2}G_{ij}^2 - \frac{1}{2}K_{ij}^2 - K_{ij}(G_{ij}-K_{ij}) \\ & = \frac{1}{2}\|\mathbf{G}-\mathbf{K}\|_F^2 = \frac{1}{2}tr\left((\mathbf{G}-\mathbf{K})^\top(\mathbf{G}-\mathbf{K})\right) \\ & = \frac{1}{2}tr\left(\mathbf{K}^\top\mathbf{K} + \mathbf{G}^\top\mathbf{G} - 2\mathbf{K}^\top\mathbf{G}\right) \end{aligned} \tag{9}$$

Thus, the BBS problem with $\phi(x) = x^2/2$ is equivalent to

$$\begin{aligned} \min_{\mathbf{G}} \quad & tr\left(\mathbf{G}^\top\mathbf{G} - 2\mathbf{K}^\top\mathbf{G}\right) \\ s.t. \quad & \mathbf{G} \geqslant 0, \ \ \mathbf{G} = \mathbf{G}^\top, \ \ \mathbf{G1} = \mathbf{1} \end{aligned} \tag{10}$$

Problem (10) is a *Quadratic Programming* program [2], [16] and can be solved by standard methods such as the *interior point* algorithm. Here, we adopt a simple cyclic constraint projection approach, known as the *Dykstra algorithm* [10]. First we split the constraints into two sets $\mathcal{C}_1$ and $\mathcal{C}_2$:

$$\mathcal{C}_1: \quad \{\mathbf{G}|\mathbf{G} = \mathbf{G}^\top, \ \mathbf{G1} = \mathbf{1}\} \tag{11}$$
$$\mathcal{C}_2: \quad \{\mathbf{G}|\mathbf{G} \geqslant 0\} \tag{12}$$

where $\mathcal{C}_1$ defines an affine set and $\mathcal{C}_2$ defines a convex set.

For the constraint set $\mathcal{C}_1$, we need to solve[2]

$$\begin{aligned} \min_{\mathbf{G}} \quad & tr\left(\mathbf{G}^\top\mathbf{G} - 2\mathbf{K}^\top\mathbf{G}\right) \\ s.t. \quad & \mathbf{G} = \mathbf{G}^\top, \ \mathbf{G1} = \mathbf{1}, \end{aligned} \tag{13}$$

for which we first introduce a Lagrangian function

$$\begin{aligned} \mathcal{L}(\mathbf{G}) \ = \ & tr\left(\mathbf{G}^\top\mathbf{G} - 2\mathbf{K}^\top\mathbf{G}\right) - \boldsymbol{\mu}_1^\top(\mathbf{G1}-\mathbf{1}) \\ & -\boldsymbol{\mu}_2^\top(\mathbf{G}^\top\mathbf{1}-\mathbf{1}) \end{aligned} \tag{14}$$

where $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathbb{R}^{n\times 1}$ are Lagrangian multipliers. By the constraint $\mathbf{G} = \mathbf{G}^\top$, we know $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu}$. Thus

$$\nabla_{\mathbf{G}}\mathcal{L}(\mathbf{G}) = 2(\mathbf{G}-\mathbf{K}) - \boldsymbol{\mu}\mathbf{1}^\top - \mathbf{1}\boldsymbol{\mu}^\top \tag{15}$$

Setting $\nabla_{\mathbf{G}}\mathcal{L}(\mathbf{G}) = 0$ yields

$$\mathbf{G} = \mathbf{K} + \frac{1}{2}\boldsymbol{\mu}\mathbf{1}^\top + \frac{1}{2}\mathbf{1}\boldsymbol{\mu}^\top \tag{16}$$

Since $\mathbf{G}$ must satisfy the constraint $\mathbf{G1} = \mathbf{1}$, we can right-multiply $\mathbf{1}$ on the both sides of Eq. (16) as

$$\mathbf{1} = \mathbf{G1} = \mathbf{K1} + \frac{n}{2}\boldsymbol{\mu} + \frac{1}{2}\mathbf{1}\mathbf{1}^\top\boldsymbol{\mu} \tag{17}$$

[2]It may be also formulated as an instance of the *Least Norm* problem [2].

Then we obtain

$$\boldsymbol{\mu} = 2\left(n\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)^{-1}(\mathbf{I} - \mathbf{K})\,\mathbf{1} \tag{18}$$

By making use of the *Woodbury formula* [11], we obtain

$$\left(n\mathbf{I} + \mathbf{1}\mathbf{1}^\top\right)^{-1} = \frac{1}{n}\left(\mathbf{I} - \frac{1}{2n}\mathbf{1}\mathbf{1}^\top\right) \tag{19}$$

We can then write the solution in a closed form:

$$\mathbf{G} = \mathbf{K} + \left(\frac{1}{n}\mathbf{I} - \frac{1}{n}\mathbf{K} + \frac{\mathbf{1}\mathbf{1}^\top\mathbf{K}}{n^2}\right)\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{K} \tag{20}$$

For the constraint set $\mathcal{C}_2$, we need to solve another optimization problem:

$$\begin{array}{ll} \min_{\mathbf{G}} & \frac{1}{2}\|\mathbf{G} - \mathbf{K}\|_F^2 \\ s.t. & \mathbf{G} \succcurlyeq 0 \end{array} \tag{21}$$

whose solution is simply

$$\mathbf{G} = \mathbf{K}^+ \tag{22}$$

where $\mathbf{K}^+$ denotes the positive part of $\mathbf{K}$.

The overall algorithm of BBS with $\phi(x) = x^2/2$ is summarized in Alg. 1. The total computational complexity of Alg. 1 is $O(Tn^2)$ with T being the number of iterations needed for the algorithm to converge.

Algorithm 1 BBS WITH $\phi(x) = x^2/2$

Require: An initial similarity matrix $\mathbf{K}$

1: $t = 0$, $\mathbf{G}^{(t)} = \mathbf{K}$.
2: **repeat**
3: $\quad t \leftarrow t + 1$
4: $\quad \mathbf{G}^{(t)} \leftarrow \left[\mathbf{G}^{(t-1)} + \frac{1}{n}\left(\mathbf{I} - \mathbf{G}^{(t-1)} + \frac{\mathbf{1}\mathbf{1}^\top\mathbf{G}^{(t-1)}}{n}\right)\mathbf{1}\mathbf{1}^\top - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{G}^{(t-1)}\right]^+$
5: **until** Some convergence condition is satisfied
6: Output $\mathbf{G}^{(t)}$

B. $\phi(x) = x\log x - x$

$$\begin{aligned} D_\phi(\mathbf{G}, \mathbf{K}) &= \sum_{ij} D_\phi(G_{ij}, K_{ij}) \\ &= \sum_{ij} G_{ij}\log\frac{G_{ij}}{K_{ij}} + K_{ij} - G_{ij} = KL(\mathbf{G}\|\mathbf{K}) \end{aligned} \tag{23}$$

The BBS problem becomes

$$\begin{array}{ll} \min_{\mathbf{G}} & KL(\mathbf{G}\|\mathbf{K}) \\ s.t. & \mathbf{G} \succcurlyeq 0,\ \mathbf{G} = \mathbf{G}^\top,\ \mathbf{G}\mathbf{1} = \mathbf{1} \end{array} \tag{24}$$

We construct the following Lagrangian

$$\mathcal{L}(\mathbf{G}) = KL(\mathbf{G}\|\mathbf{K}) - \boldsymbol{\mu}_1^\top(\mathbf{G}^\top\mathbf{1} - \mathbf{1}) - \boldsymbol{\mu}_2^\top(\mathbf{G}\mathbf{1} - \mathbf{1}), \tag{25}$$

where we drop the constraint $\mathbf{G} \succcurlyeq 0$ for the time being, and we will later show it is automatically satisfied. Therefore

$$\nabla_G\mathcal{L}(\mathbf{G}) = \log\mathbf{G} - \log\mathbf{K} - \boldsymbol{\mu}_1\mathbf{1}^\top - \mathbf{1}\boldsymbol{\mu}_2^\top \tag{26}$$

where log represents the elementwise logarithm. Setting $\nabla_G\mathcal{L}(\mathbf{G}) = 0$ yields

$$\log G_{ij} - \log K_{ij} - \mu_{1i} - \mu_{2j} = 0 \tag{27}$$

Thus the solution satisfies

$$G_{ij} = e^{\mu_{1i}}K_{ij}e^{\mu_{2j}} \tag{28}$$

Next, we define the following two vectors

$$\boldsymbol{\pi}_1 = \left[e^{\mu_{11}}, e^{\mu_{12}}, \cdots, e^{\mu_{1n}}\right]^{\mathrm{T}} \in \mathbb{R}^{n\times 1} \tag{29}$$

$$\boldsymbol{\pi}_2 = \left[e^{\mu_{21}}, e^{\mu_{22}}, \cdots, e^{\mu_{2n}}\right]^{\mathrm{T}} \in \mathbb{R}^{n\times 1} \tag{30}$$

and two diagonal matrices $\mathrm{diag}(\boldsymbol{\pi}_1) \in \mathbb{R}^{n\times n}$, $\mathrm{diag}(\boldsymbol{\pi}_2) \in \mathbb{R}^{n\times n}$. This way, we can express the solution to be

$$\mathbf{G} = \mathrm{diag}(\boldsymbol{\pi}_1) \times \mathbf{K} \times \mathrm{diag}(\boldsymbol{\pi}_2) \tag{31}$$

As $\mathbf{G}$ is symmetric, we know $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu}$ and $\boldsymbol{\pi}_1 = \boldsymbol{\pi}_2 = \boldsymbol{\pi}$. By comparing with the *Sinkhorn-Knopp* Theorem, we can immediately see that the BBS algorithm with $\phi(x) = x\log x - x$ actually recovers the symmetric SK algorithm, and $\mathrm{diag}(\boldsymbol{\pi})$ is used for scaling $\mathbf{K}$ to be bi-stochastic.

We should mention that, it appears that the fact that the symmetric SK algorithm minimizes the KL divergence was essentially discovered in statistics [4], [19].

III. Experiments

A. Data Sets

Table I summarizes the data sets used in our experiments:

- **MNIST**[3]: We randomly sampled 6000 data points from the original training set. We also created a smaller data set, **MNIST (0-4)**, using digits 0, 1, 2, 3, 4.
- **ISOLET**[4]: We took the original UCI training set and divided it into three smaller data sets so that the number of classes (clusters) for each set is not too large.
- **LETTER**[5]: We divided the original data into five sets.
- **NEWS20**[6]: The test set from the LibSVM site.
- **OPTDIGIT**[7]: We combined the original (UCI) training and test sets, as this data set is not too large.
- **PENDIGIT**[8]: The original (UCI) training set.
- **SATIMAGE**[9]: The original (UCI) training set.
- **SHUTTLE**[10]: The test set from the LibSVM site.
- **VEHICLE**[11]: The version from the LibSVM site.
- **ZIPCODE**[12]: We used the training set and also constructed a smaller data set using digits 0, 1, 2, 3, 4.

[3] http://yann.lecun.com/exdb/mnist/

[4] http://archive.ics.uci.edu/ml/machine-learning-databases/isolet/isolet1+2+3+4.data.Z

[5] http://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/letter-recognition.data

[6] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/news20.t.scale.bz2

[7] http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[8] http://archive.ics.uci.edu/ml/machine-learning-databases/pendigits/pendigits.tra

[9] http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/satimage/sat.trn

[10] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/shuttle.scale.t

[11] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/vehicle.scale

[12] http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/zip.train.gz

Table I
DATA SETS

Data Set	# Samples (n)	# Dimensions (d)	# Classes (k)
MNIST	6000	784	10
MNIST (0-4)	3031	784	5
ISOLET (A-I)	2158	617	9
ISOLET (J-R)	2160	617	9
ISOLET (S-Z)	1920	617	8
LETTER (A-E)	3864	16	5
LETTER (F-J)	3784	16	5
LETTER (K-O)	3828	16	5
LETTER (P-T)	3888	16	5
LETTER (U-Z)	4636	16	6
NEWS20	3993	62060	20
OPTDIGIT	5620	64	10
PENDIGIT	7494	16	10
SATIMAGE	4465	36	6
SHUTTLE	14500	9	7
VEHICLE	846	18	4
ZIPCODE	7291	256	10
ZIPCODE (0-4)	4240	256	5

B. Experiment Procedure

For all data sets, we always normalized each data point (vector) to have a unit l_2 norm, and we always used the Gaussian kernel Eq. (4) to form the initial similarity matrix $\mathbf{K}$. For the tuning parameter γ in Eq. (4), we experimented with $\gamma \in \{1024, 256, 64, 32, 16, 8, 4, 2, 1, 0.5, 0.25\}$.

We ran the BBS algorithm with $\phi(x) = x^2/2$ for 1000 iterations at each γ. We also ran the SK algorithm (i.e., BBS with $\phi(x) = x \log x - x$) for 1000 iterations at each γ.

We eventually used spectral clustering [17], [3], [8], [15] to evaluate the quality of the produced bi-stochastic matrices. In particular, we used the procedure described in [15]. That is, we computed the top-k eigenvectors of the bi-stochastic matrix to form a new $n \times k$ matrix and normalized each row to have a unit l_2 norm. Denote the resulting new "data matrix" by $\mathbf{Z}$. We then used Matlab *kmeans* function:

```
kmeans(Z, k,'MaxIter',1000,'EmptyAction','singleton')
```

We ran *kmeans* 100 times and reported both the *average* and *maximum* clustering results.

However, we would like to first introduce two measures that may allow us to directly assess the quality of the bi-stochastic matrices independent of the clustering algorithms.

C. Quality Measurements of the Bi-Stochastic Matrices

After we have generated a (hopefully) bi-stochastic matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ from a similarity matrix $\mathbf{K}$, we can compute:

$$M_B = \frac{1}{n} \sum_{i=1}^{n} \left| \sum_{j=1}^{n} P_{ij} - 1 \right|, \tag{32}$$

$$M_C = \sum_{c=1}^{k} \frac{1}{n} \sum_{i=1}^{n} \left| \sum_{j=1, x_j \in \pi_c}^{n} P_{ij} - 1 \right| \tag{33}$$

Basically, M_B measures how far $\mathbf{P}$ is from being a bi-stochastic matrix, and M_C roughly measures the potential of producing good clustering results. Lower values of M_B and M_C are more desirable. We use the M_C measure because it is independent of the specific clustering algorithms.

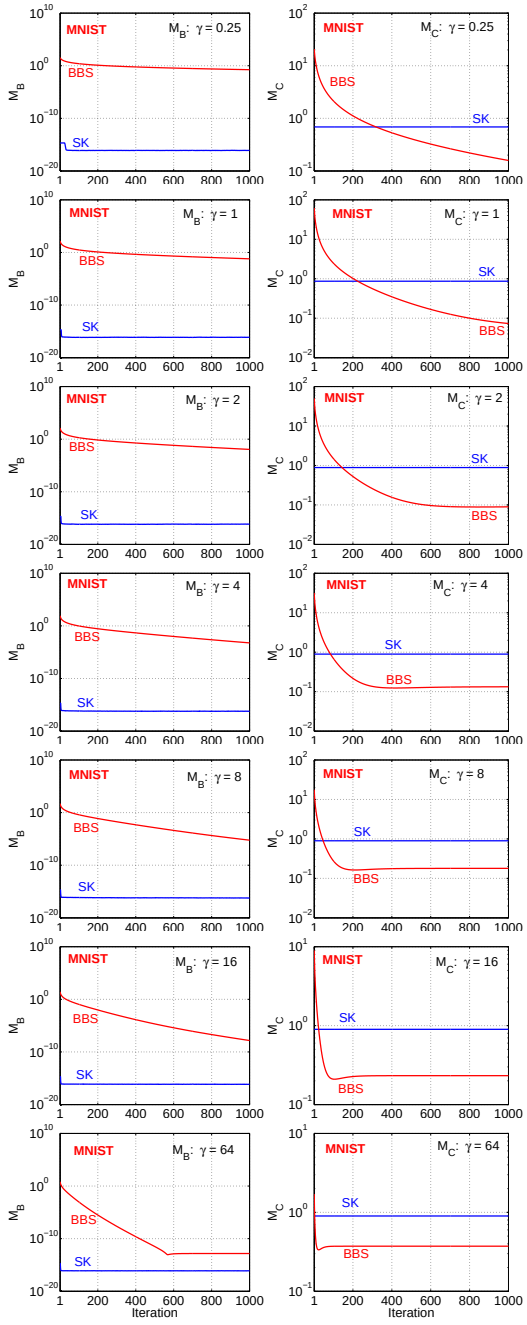

Figure 1. Quality measurements M_B (32) and M_C (33) (lower is better), on **MNIST** data, for up to 1000 iterations. γ is the kernel tuning parameter in Eq. (4). "BBS" labels the curves produced by the BBS algorithm with $\phi(x) = x^2/2$ (i.e., Alg. 1) and "SK" the curves by the SK algorithm.

Fig. 1 presents the quality measurements on the **MNIST** data set, for a wide range of γ values. Fig. 2 presents the measurements on a variety of data sets for $\gamma = 1$:

- In terms of M_B, the SK algorithm performs well in producing good bi-stochastic matrices.
- In terms of M_C, the BBS algorithm using the Euclidian distance (Alg. 1) has noticeably better potential of producing good clustering results than the SK algorithm.

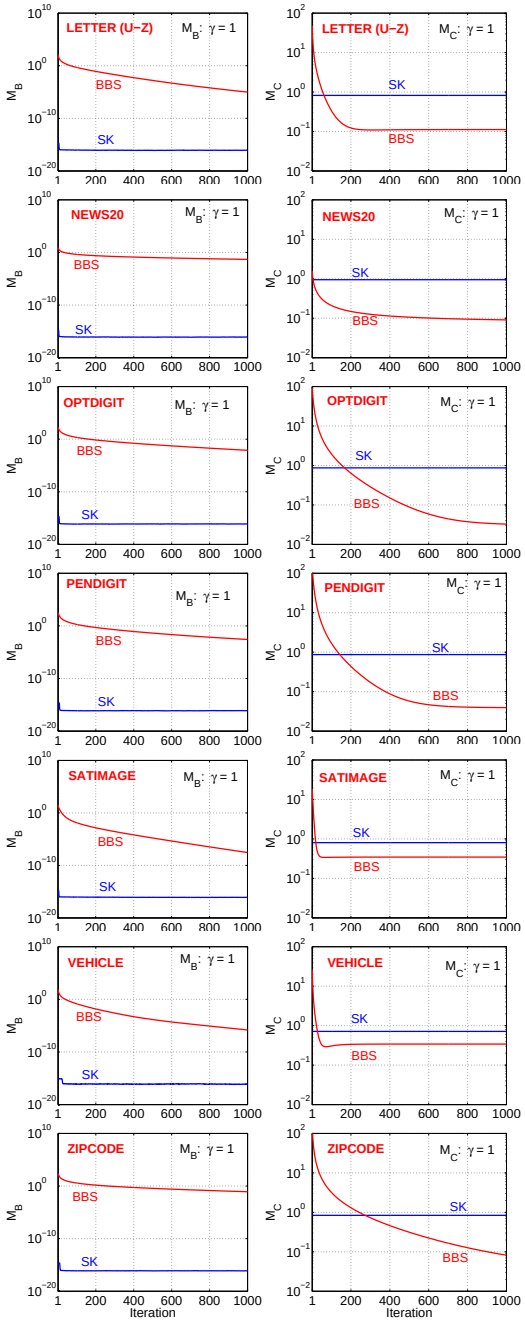

Figure 2. Quality measurements, M_B, M_C, on a variety of data sets.

D. Comparing Clustering Results

We ultimately rely on the standard clustering procedure, e.g., [15], to assess clustering quality. Tables II to V provide the results for **BBS** (Alg. 1), **SK**, and three other methods:

- **K-means**: We directly used the original data sets (after normalizing each data point to have a unit l_2 norm) and ran Matlab *kmeans* 100 times.
- **RA**: We ran spectral clustering directly on the similarity matrix $\mathbf{K}$ (4). It is called *Ratio Association* [6].
- **Ncut**: We ran spectral clustering on the normalized similarity matrix $\widetilde{\mathbf{K}} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$, as in Eq. (5).

We report the clustering results on two metrics:

1) **Clustering Accuracy**:

$$Accuracy = \frac{1}{n}\max\left(\sum_{\pi_i,\hat{\pi}_j}|\pi_i \cap \hat{\pi}_j|\right), \qquad (34)$$

where $\hat{\pi}_j$ denotes the j-th cluster in the output, π_i is the true i-th class, and $|\pi_i \cap \hat{\pi}_j|$ is the number of data points from the i-th class are assigned to j-th cluster.

2) **Normalized Mutual Information** (***NMI***) [21]:

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}|\pi_i \cap \hat{\pi}_j|\log\left(\frac{n\cdot|\pi_i\cap\hat{\pi}_j|}{|\pi_i|\cdot|\hat{\pi}_j|}\right)}{\sqrt{\left(\sum_{i=1}^{k}|\pi_i|\log\frac{|\pi_i|}{n}\right)\left(\sum_{j=1}^{k}|\hat{\pi}_j|\log\frac{|\hat{\pi}_j|}{n}\right)}} \qquad (35)$$

We still need to address two more issues:

- For each case, we always ran *kmeans* 100 times. We report both the *average* and *maximum* measures of clustering quality (*Accuracy* and *NMI*). In practice, the *maximum* clustering performance may be quite attainable by tuning and running *kmeans* many times with different (random) initial starts.
- For **RA, Ncut, SK** and **BBS**, we experimented with the similarity matrices $\mathbf{K}$ (4) generated from a series of γ values (from 0.25 to 1024). Tables II to V report the best results among all γ's. Again, the rationale is that, in practice, the best performance may be attainable by careful tuning. In addition, we believe it is also informative to present the results for all γ values, as in the Appendix; although due to the space limit, we could not present the results for all data sets.

Tables II to V demonstrate that, for many data sets, BBS (Alg. 1) can achieve considerably better clustering results than other methods, especially when evaluated using *maximum accuracy* and *maximum NMI*.

Table II
AVERAGE ACCURACY

Data	K-means	RA	Ncut	SK	BBS
MNIST	0.536	0.552	0.545	0.542	**0.633**
MNIST (0-4)	0.744	0.722	0.722	0.721	**0.805**
ISOLET (A-I)	0.621	**0.737**	0.735	0.709	0.713
ISOLET (J-R)	0.662	0.706	0.705	0.702	**0.708**
ISOLET (S-Z)	0.703	**0.787**	0.739	0.742	0.773
LETTER (A-E)	0.462	0.516	0.516	0.513	**0.539**
LETTER (F-J)	0.514	0.490	0.492	0.495	**0.619**
LETTER (K-O)	0.390	0.474	0.473	0.470	**0.502**
LETTER (P-T)	0.426	0.554	0.554	**0.555**	0.554
LETTER (U-Z)	0.467	0.511	**0.517**	0.512	0.505
NEWS20	0.273	0.244	0.245	0.244	**0.378**
OPTDIGIT	0.750	0.791	0.767	0.762	**0.848**
PENDIGIT	0.703	0.730	0.732	0.733	**0.756**
SATIMAGE	0.607	0.565	0.569	0.573	**0.617**
SHUTTLE	0.464	0.384	0.448	0.453	**0.647**
VEHICLE	0.366	0.389	0.371	0.374	**0.409**
ZIPCODE	0.650	0.678	0.678	0.674	**0.747**
ZIPCODE (0-4)	0.760	0.686	0.680	0.684	**0.908**

Table III
AVERAGE NMI

$Data$	K-means	RA	Ncut	SK	BBS
MNIST	0.523	0.517	0.524	0.507	**0.711**
MNIST (0-4)	0.670	0.638	0.652	0.667	**0.850**
ISOLET (A-I)	0.711	0.756	0.755	0.746	**0.808**
ISOLET (J-R)	0.762	0.760	0.760	0.745	**0.832**
ISOLET (S-Z)	0.790	0.803	0.788	0.781	**0.843**
LETTER (A-E)	0.320	0.348	0.350	0.347	**0.397**
LETTER (F-J)	0.379	0.337	0.342	0.352	**0.469**
LETTER (K-O)	0.265	0.260	0.262	0.254	**0.379**
LETTER (P-T)	0.263	0.371	0.372	0.373	**0.417**
LETTER (U-Z)	0.344	0.397	0.403	0.399	**0.437**
NEWS20	0.319	0.241	0.241	0.238	**0.422**
OPTDIGIT	0.728	0.748	0.725	0.709	**0.874**
PENDIGIT	0.691	0.693	0.693	0.705	**0.776**
SATIMAGE	0.549	0.473	0.491	0.494	**0.603**
SHUTTLE	0.496	0.396	0.429	0.448	**0.542**
VEHICLE	0.116	0.108	0.124	0.123	**0.168**
ZIPCODE	0.631	0.625	0.625	0.624	**0.815**
ZIPCODE (0-4)	0.716	0.703	0.712	0.700	**0.913**

Table IV
MAXIMUM ACCURACY

$Data$	K-means	RA	Ncut	SK	BBS
MNIST	0.588	0.608	0.603	0.603	**0.738**
MNIST (0-4)	0.853	0.756	0.790	0.827	**0.960**
ISOLET (A-I)	0.738	0.798	0.798	0.798	**0.819**
ISOLET (J-R)	0.760	0.750	0.746	0.746	**0.781**
ISOLET (S-Z)	0.861	0.846	0.870	0.794	**0.933**
LETTER (A-E)	0.518	0.589	0.589	0.589	**0.595**
LETTER (F-J)	0.590	0.584	0.584	0.546	**0.649**
LETTER (K-O)	0.463	0.487	0.500	0.510	**0.560**
LETTER (P-T)	0.496	0.604	0.556	0.556	**0.621**
LETTER (U-Z)	0.532	0.567	0.558	0.558	**0.585**
NEWS20	0.284	0.264	0.262	0.259	**0.419**
OPTDIGIT	0.875	0.814	0.824	0.801	**0.911**
PENDIGIT	0.778	0.795	0.794	0.820	**0.857**
SATIMAGE	0.632	0.582	0.588	0.590	**0.639**
SHUTTLE	0.598	0.474	0.506	0.510	**0.861**
VEHICLE	0.402	0.395	0.382	0.382	**0.479**
ZIPCODE	0.756	0.740	0.739	0.731	**0.897**
ZIPCODE (0-4)	0.891	0.813	0.808	0.809	**0.991**

Table V
MAXIMUM NMI

$Data$	K-means	RA	Ncut	SK	BBS
MNIST	0.567	0.542	0.567	0.541	**0.741**
MNIST (0-4)	0.696	0.641	0.659	0.680	**0.897**
ISOLET (A-I)	0.788	0.779	0.781	0.770	**0.862**
ISOLET (J-R)	0.812	0.800	0.792	0.786	**0.883**
ISOLET (S-Z)	0.874	0.843	0.878	0.806	**0.897**
LETTER (A-E)	0.392	0.422	0.422	0.422	**0.468**
LETTER (F-J)	0.435	0.412	0.412	0.406	**0.524**
LETTER (K-O)	0.306	0.288	0.301	0.298	**0.430**
LETTER (P-T)	0.378	0.422	0.375	0.377	**0.499**
LETTER (U-Z)	0.395	0.445	0.437	0.437	**0.502**
NEWS20	0.336	0.254	0.254	0.252	**0.433**
OPTDIGIT	0.786	0.758	0.755	0.750	**0.897**
PENDIGIT	0.718	0.717	0.719	0.734	**0.826**
SATIMAGE	0.627	0.483	0.511	0.511	**0.623**
SHUTTLE	0.563	0.516	0.507	0.489	**0.705**
VEHICLE	0.172	0.125	0.150	0.144	**0.234**
ZIPCODE	0.665	0.652	0.649	0.645	**0.871**
ZIPCODE (0-4)	0.755	0.705	0.712	0.706	**0.964**

IV. RELATIONSHIP TO k-MEANS

It is beneficial to gain some intuitive understanding on why the BBS algorithm with $\phi(x) = x^2/2$ (i.e., Alg. 1) can perform well in clustering. We show that it is closely related to various relaxed k-means algorithms.

The k-means clustering aims to minimize the objective

$$J_1 = \sum\nolimits_{c=1}^{k} \sum\nolimits_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mu_c\|^2 \tag{36}$$

where μ_c is the mean of cluster π_c. Some algebra can show that the minimizing J_1 is equivalent to minimizing J_2:

$$J_2 = -tr\left(\widetilde{\mathbf{F}}^\top \mathbf{X}\mathbf{X}^\top \widetilde{\mathbf{F}}\right) \tag{37}$$

where $\widetilde{\mathbf{F}}$ is the scaled partition matrix introduced at the beginning of the paper and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^\mathrm{T} \in \mathbb{R}^{n\times d}$ is the data matrix. Let $\mathbf{G} = \widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top$ and $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$. Then $J_2 = -tr\left(\mathbf{K}\mathbf{G}\right)$, which in fact can be viewed as a special case of the objective of BBS defined in Eq. (10): $tr\left(\mathbf{G}^\top\mathbf{G} - 2\mathbf{K}^\top\mathbf{G}\right)$, because the term $tr\left(\mathbf{G}^\top\mathbf{G}\right)$ can be treated as a constant in this case:

$$tr\left(\mathbf{G}^\top\mathbf{G}\right) = tr\left(\widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top\widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top\right) = tr\left(\widetilde{\mathbf{F}}\widetilde{\mathbf{F}}^\top\right) = tr(\mathbf{G}) = k$$

In addition, $\mathbf{K} = \mathbf{X}\mathbf{X}^\mathrm{T}$ is the linear kernel, which may be replaced by more flexible kernels, e.g., Eq. (4) as we use.

There are more than one way to formulate the relaxed k-means algorithm. For example,

$$\begin{aligned} \min\nolimits_{\mathbf{G}} \quad & D_\phi(\mathbf{G}, \mathbf{K}), \qquad (\text{where } \ \phi(x) = x^2) && (38)\\ s.t. \quad & \mathbf{G} \geqslant 0, \ \ \mathbf{G} = \mathbf{G}^\top, \ \ \mathbf{G}\mathbf{1} = \mathbf{1}, \\ & \mathbf{G}^2 = \mathbf{G}, \ \ tr(\mathbf{G}) = k, && (39) \end{aligned}$$

which is quite similar to our formulation of the BBS problem with the Euclidian distance. Our formulation discards the constraints (39) and hence its optimization task is easier.

V. EXTENSION: MULTIPLE BBS (MBBS)

Our detailed experiments reported in the Appendix illustrate that the clustering performance of the BBS algorithm (as well as other algorithms), to an extent, depends on the initial similarity matrix $\mathbf{K}$. This section extends BBS to combine the power of multiple input similarity matrices, e.g., a series of kernel matrices (4) using different γ values, to boost the performance. We name this scheme *Multiple BBS* or *MBBS*. This is in spirit related to *cluster ensemble* [21] and *Generalized Cluster Aggregation* [22].

Suppose we have m similarity matrices $\{\mathbf{K}_{(i)}\}_{i=1}^m$. We would like to obtain a bi-stochastic similarity matrix $\mathbf{G}$ by solving the following optimization problem:

$$\begin{aligned} \min\nolimits_{\mathbf{G},\boldsymbol{\alpha}} \quad & \sum\nolimits_{i=1}^{m} \alpha_i D_\phi\left(\mathbf{G}, \mathbf{K}_{(i)}\right) + \lambda\Omega(\boldsymbol{\alpha}) && (40)\\ s.t. \quad & \mathbf{G} \geqslant 0, \ \ \mathbf{G} = \mathbf{G}^\top, \ \ \mathbf{G}\mathbf{1} = \mathbf{1}, \\ & \forall\ i, \ \alpha_i \geqslant 0, \ \ \sum\nolimits_{i=1}^{m} \alpha_i = 1 \end{aligned}$$

We constrain the weight coefficients $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^m$ to be in a *simplex*. $\Omega(\boldsymbol{\alpha})$ is some regularizer to avoid trivial solutions.

There are two groups of variables $\boldsymbol{\alpha}$ and $\mathbf{G}$. Although the problem (40) is not jointly convex, it is convex with respect to one group of variables with the other group being fixed. Thus, it is reasonable to apply *block coordinate descent* [1].

A. Fix $\boldsymbol{\alpha}$, Solve $\mathbf{G}$

At the t-th iteration, if $\boldsymbol{\alpha}$ is fixed to be $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(t-1)}$, the problem (40) becomes

$$\begin{array}{ll} \min_{\mathbf{G}} & \sum_{i=1}^{m} \alpha_i^{(t-1)} D_\phi \left(\mathbf{G}, \mathbf{K}_{(i)}\right) \\ s.t. & \mathbf{G} \geqslant 0, \quad \mathbf{G} = \mathbf{G}^\top, \quad \mathbf{G1} = \mathbf{1}. \end{array} \tag{41}$$

Note that $\Omega(\boldsymbol{\alpha})$ is irrelevant at this point. This is similar to problem (7) except for the summation form in the objective. The solution procedures are consequently also similar.

Here we assume $\phi(x) = x^2/2$ for the illustration purpose.

$$\begin{aligned} &\sum_{i=1}^{m} \alpha_i^{(t-1)} D_\phi \left(\mathbf{G}, \mathbf{K}_{(i)}\right) \\ =&\frac{1}{2} tr \left(\sum_{i=1}^{m} \alpha_i^{(t-1)} \mathbf{K}_{(i)}^\top \mathbf{K}_{(i)} + \mathbf{G}^\top \mathbf{G} - 2 \sum_{i=1}^{m} \alpha_i^{(t-1)} \mathbf{K}_{(i)}^\top \mathbf{G} \right) \end{aligned}$$

where we use the fact $\sum_{i=1}^{m} \alpha_i^{(t-1)} = 1$. As the term $\sum_{i=1}^{m} \alpha_i^{(t-1)} \mathbf{K}_{(i)}^\top \mathbf{K}_{(i)}$ is irrelevant, the problem becomes

$$\begin{array}{ll} \min_{\mathbf{G}} & tr \left(\mathbf{G}^\top \mathbf{G} - 2 \left(\sum_{i=1}^{m} \alpha_i \mathbf{K}_{(i)} \right)^\top \mathbf{G} \right) \\ s.t. & \mathbf{G} \geqslant 0, \quad \mathbf{G} = \mathbf{G}^\top, \quad \mathbf{G1} = \mathbf{1} \end{array} \tag{42}$$

which is the same as Problem (10) if we make

$$\mathbf{K} = \sum_{i=1}^{m} \alpha_i^{(t-1)} \mathbf{K}_{(i)}. \tag{43}$$

B. Fix $\mathbf{G}$, Solve $\boldsymbol{\alpha}$

When $\mathbf{G}$ is fixed with $\mathbf{G} = \mathbf{G}^{(t)}$ and for simplicity we only consider $\Omega(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|^2 = \boldsymbol{\alpha}^\top \boldsymbol{\alpha}$, the problem becomes

$$\begin{array}{ll} \min_{\mathbf{G}, \boldsymbol{\alpha}} & \sum_{i=1}^{m} \alpha_i D_\phi \left(\mathbf{G}^{(t)}, \mathbf{K}_i\right) + \lambda \|\boldsymbol{\alpha}\|^2 \\ s.t. & \forall\ i,\ \alpha_i \geqslant 0, \quad \sum_{i=1}^{m} \alpha_i = 1, \end{array} \tag{44}$$

which is a standard *Quadratic Programming* (QP) problem.

Here we will reformulate this problem to facilitate more efficient solutions. For the notational convenience, we denote $\mathbf{g}^{(t)} = (g_1^{(t)}, g_2^{(t)}, \cdots, g_m^{(t)})^\top$ with

$$g_i^{(t)} = D_\phi \left(\mathbf{G}^{(t)}, \mathbf{K}_{(i)}\right) \tag{45}$$

We first rewrite the objective of Problem (44) as

$$\boldsymbol{\alpha}^\top \mathbf{g}^{(t)} + \lambda \|\boldsymbol{\alpha}\|^2 = \left\| \sqrt{\lambda} \boldsymbol{\alpha} - \frac{1}{\sqrt{2\lambda}} \mathbf{g}^{(t)} \right\|^2 + \frac{1}{2\lambda} \left(\mathbf{g}^{(t)}\right)^\top \mathbf{g}^{(t)}$$

As $\frac{1}{2\lambda} \left(\mathbf{g}^{(t)}\right)^\top \mathbf{g}^{(t)}$ is irrelevant, (44) can be rewritten to be

$$\min_{\boldsymbol{\alpha}} \ \left\| \boldsymbol{\alpha} - \tfrac{1}{\sqrt{2\lambda}} \mathbf{g}^{(t)} \right\|^2, \quad s.t. \ \ \boldsymbol{\alpha} \geqslant 0, \ \boldsymbol{\alpha}^\top \mathbf{1} = 1, \tag{46}$$

which is an Euclidian projection problem under the simplex constraint and can be solved efficiently, e.g., [9][14].

We will report extensive experiment results of Multiple BBS in a more comprehensive technical report.

VI. Conclusions

We present **BBS** (*Bregmanian Bi-Stochastication*), a general framework for learning a bi-stochastic data similarity matrix from an initial similarity matrix, by minimizing the Bregmanian divergences such as the Euclidian distance or the KL divergence. The resultant bi-stochastic matrix can be used as input to clustering algorithms. The BBS framework is closely related to the relaxed k-means algorithms. Our extensive experiments on a wide range of public data sets demonstrate that the BBS algorithm using the Euclidian distance can often produce noticeably superior clustering results than other well-known algorithms including the **SK** algorithm and the **Ncut** algorithm.

Acknowledgement

This work is partially supported by NSF (DMS-0808864), ONR (YIP-N000140910911), and a grant from Microsoft.

References

[1] D. P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 1999.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK., 2004.

[3] P. K. Chan, D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. Computer-Aided Design*, 13:1088–1096, 1994.

[4] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[5] W. E. Deming and F. F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444, 1940.

[6] I. S. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical report, Department of Computer Science, University of Texas at Austin. TR-04-25, 2004.

[7] I. S. Dhillon and J. A. Tropp. Matrix nearness problems with bregman divergences. In *SIAM Journal on Matrix Analysis and Applications*, volume 29, pages 1120–1146, 2008.

[8] C. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 1st International Conference on Data Mining*, pages 107–114, 2001.

[9] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[10] R. Escalante and M. Raydan. Dykstra's algorithm for a constrained least-squares matrix problem. *Numerical Linear Algebra with Applications*, 3(6):459–471, 1998.

[11] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[12] A. Horn. Doubly stochastic matrices and the diagonal of a rotation matrix. *The American Journal of Mathematics*, 76:620–630, 1954.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[14] J. Liu and J. Ye. Efficient Euclidean projections in linear time. In *International Conference on Machine Learning*, pages 657–664, 2009.

[15] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[16] J. Nocedal and S. J. Wright. *Numerical Optimization (2nd ed.)*. Springer-Verlag, Berlin, New York, 2006.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[18] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.*, 21:343–348, 1967.

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

[20] F. F. Stephan. An iterative method of adjusting sample frequency tables when expected marginal totals are known. *The Annals of Mathematical Statistics*, 13(2):166–178, 1942.

[21] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[22] F. Wang, X. Wang, and T. Li. Generalized cluster aggregation. In *Proceedings of The 21st International Joint Conference on Artificial Intelligence*, pages 1279–1284, 2009.

[23] R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *Proceedings of International Conference on Computer Vision*, pages 294–301, 2005.

[24] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14*, 2001.

APPENDIX

We generated the base similarity matrix $\mathbf{K}$ using the Gaussian kernel (4) which has a tuning parameter $\gamma > 0$. The clustering performance can be, to an extent, sensitive to γ; and hence we would like to present the clustering results for γ values ranging from $2^{-2} = 0.25$ to $2^{10} = 1024$, for four algorithms: **RA, Ncut, SK**, and **BBS** (using Euclidian distance), and two performance measures: *Accuracy* and *NMI*, as defined in Eq. (34) and Eq (35), respectively.

In the tables, each entry contains the *average* and *maximum* (in parentheses) clustering results from 100 runs of the Matlab *kmeans* program. Due to the space limit, we could not present the experiments for all the data sets.

MNIST:			**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.532 (0.595)	0.533 (0.596)	0.536 (0.595)	0.553 (0.606)
256	0.536 (0.596)	0.537 (0.594)	0.539 (0.594)	0.566 (0.617)
64	0.541 (0.608)	0.540 (0.596)	0.530 (0.600)	0.581 (0.642)
32	0.538 (0.596)	0.537 (0.596)	0.537 (0.596)	0.612 (0.665)
16	0.537 (0.606)	0.532 (0.598)	0.538 (0.600)	0.630 (0.690)
8	0.539 (0.594)	0.536 (0.594)	0.541 (0.598)	0.622 (0.702)
4	0.537 (0.600)	0.542 (0.596)	0.538 (0.598)	0.630 (0.715)
2	0.540 (0.596)	0.542 (0.599)	0.542 (0.603)	0.633 (0.720)
1	0.548 (0.597)	0.545 (0.603)	0.536 (0.597)	0.621 (0.738)
0.5	0.552 (0.601)	0.535 (0.599)	0.521 (0.597)	0.463 (0.519)
0.25	0.527 (0.590)	0.514 (0.595)	0.493 (0.558)	0.451 (0.472)
MNIST:			**NMI**	
1024	0.473 (0.512)	0.475 (0.511)	0.476 (0.511)	0.491 (0.522)
256	0.475 (0.512)	0.475 (0.510)	0.476 (0.510)	0.530 (0.564)
64	0.479 (0.512)	0.478 (0.510)	0.474 (0.510)	0.593 (0.620)
32	0.478 (0.512)	0.476 (0.506)	0.476 (0.511)	0.626 (0.648)
16	0.475 (0.504)	0.475 (0.511)	0.476 (0.511)	0.651 (0.675)
8	0.479 (0.511)	0.476 (0.511)	0.477 (0.511)	0.668 (0.692)
4	0.475 (0.507)	0.479 (0.512)	0.479 (0.513)	0.686 (0.705)
2	0.481 (0.516)	0.483 (0.516)	0.485 (0.518)	0.700 (0.724)
1	0.494 (0.524)	0.493 (0.521)	0.492 (0.511)	0.711 (0.741)
0.5	0.508 (0.526)	0.510 (0.528)	0.507 (0.541)	0.467 (0.489)
0.25	0.517 (0.542)	0.524 (0.567)	0.500 (0.525)	0.386 (0.400)
MNIST 0-4:			**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.721 (0.721)	0.719 (0.721)	0.720 (0.721)	0.722 (0.724)
256	0.720 (0.721)	0.719 (0.721)	0.719 (0.721)	0.673 (0.780)
64	0.720 (0.722)	0.722 (0.722)	0.721 (0.722)	0.762 (0.896)
32	0.721 (0.722)	0.721 (0.722)	0.721 (0.721)	0.762 (0.879)
16	0.722 (0.722)	0.721 (0.721)	0.720 (0.721)	0.757 (0.885)
8	0.721 (0.721)	0.718 (0.721)	0.715 (0.719)	0.760 (0.881)
4	0.718 (0.718)	0.716 (0.716)	0.714 (0.715)	0.777 (0.907)
2	0.703 (0.707)	0.698 (0.703)	0.692 (0.692)	0.762 (0.921)
1	0.675 (0.705)	0.675 (0.708)	0.675 (0.709)	0.805 (0.960)
0.5	0.661 (0.756)	0.658 (0.757)	0.645 (0.742)	0.438 (0.519)
0.25	0.506 (0.584)	0.708 (0.790)	0.721 (0.827)	0.532 (0.565)
MNIST 0-4:			**NMI**	
1024	0.572 (0.575)	0.571 (0.575)	0.572 (0.575)	0.577 (0.579)
256	0.572 (0.576)	0.571 (0.575)	0.571 (0.575)	0.635 (0.637)
64	0.573 (0.576)	0.575 (0.576)	0.574 (0.576)	0.741 (0.779)
32	0.576 (0.576)	0.575 (0.576)	0.575 (0.576)	0.764 (0.789)
16	0.577 (0.577)	0.577 (0.577)	0.576 (0.576)	0.785 (0.811)
8	0.579 (0.579)	0.578 (0.579)	0.577 (0.577)	0.800 (0.823)
4	0.585 (0.585)	0.583 (0.583)	0.580 (0.582)	0.815 (0.837)
2	0.589 (0.590)	0.587 (0.589)	0.590 (0.590)	0.813 (0.853)
1	0.614 (0.614)	0.609 (0.611)	0.602 (0.603)	0.850 (0.897)
0.5	0.638 (0.641)	0.632 (0.640)	0.636 (0.648)	0.346 (0.376)
0.25	0.560 (0.570)	0.652 (0.659)	0.667 (0.680)	0.418 (0.454)
ISOLET A-I:			**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.695 (0.769)	0.688 (0.768)	0.692 (0.769)	0.691 (0.768)
256	0.694 (0.769)	0.697 (0.769)	0.695 (0.769)	0.681 (0.745)
64	0.692 (0.769)	0.688 (0.769)	0.692 (0.769)	0.657 (0.743)
32	0.699 (0.769)	0.702 (0.769)	0.693 (0.769)	0.658 (0.740)
16	0.687 (0.769)	0.684 (0.769)	0.689 (0.769)	0.682 (0.813)
8	0.697 (0.769)	0.700 (0.769)	0.694 (0.769)	0.679 (0.812)
4	0.695 (0.769)	0.692 (0.769)	0.684 (0.768)	0.674 (0.812)
2	0.732 (0.781)	0.735 (0.777)	0.694 (0.769)	0.678 (0.819)
1	0.737 (0.798)	0.726 (0.798)	0.709 (0.798)	0.678 (0.779)
0.5	0.715 (0.783)	0.705 (0.743)	0.700 (0.748)	0.713 (0.787)
0.25	0.677 (0.747)	0.687 (0.750)	0.552 (0.629)	0.590 (0.639)
ISOLET A-I:			**NMI**	
1024	0.677 (0.746)	0.675 (0.746)	0.673 (0.745)	0.680 (0.745)
256	0.681 (0.745)	0.683 (0.745)	0.682 (0.746)	0.727 (0.762)
64	0.679 (0.745)	0.677 (0.745)	0.675 (0.745)	0.720 (0.773)
32	0.681 (0.746)	0.682 (0.745)	0.677 (0.746)	0.731 (0.785)
16	0.678 (0.749)	0.675 (0.745)	0.680 (0.745)	0.762 (0.822)
8	0.686 (0.748)	0.684 (0.748)	0.678 (0.746)	0.773 (0.826)
4	0.690 (0.754)	0.684 (0.747)	0.675 (0.742)	0.773 (0.832)
2	0.732 (0.756)	0.723 (0.749)	0.696 (0.742)	0.771 (0.832)
1	0.756 (0.779)	0.751 (0.777)	0.745 (0.769)	0.762 (0.791)
0.5	0.749 (0.777)	0.755 (0.775)	0.746 (0.770)	0.808 (0.862)
0.25	0.722 (0.770)	0.743 (0.781)	0.624 (0.655)	0.586 (0.614)

	ISOLET J-R:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.700 (0.747)	0.6990 (0.741)	0.702 (0.746)	0.703 (0.739)
256	0.704 (0.745)	0.7010 (0.742)	0.697 (0.741)	0.695 (0.734)
64	0.698 (0.741)	0.7050 (0.746)	0.699 (0.740)	0.678 (0.743)
32	0.702 (0.741)	0.7010 (0.739)	0.699 (0.741)	0.683 (0.745)
16	0.706 (0.741)	0.7010 (0.739)	0.702 (0.742)	0.679 (0.768)
8	0.702 (0.740)	0.7050 (0.742)	0.702 (0.742)	0.708 (0.776)
4	0.701 (0.731)	0.6950 (0.730)	0.699 (0.731)	0.685 (0.779)
2	0.701 (0.734)	0.6940 (0.734)	0.693 (0.731)	0.667 (0.781)
1	0.697 (0.736)	0.6940 (0.735)	0.690 (0.731)	0.661 (0.778)
0.5	0.702 (0.750)	0.6800 (0.741)	0.672 (0.731)	0.675 (0.777)
0.25	0.701 (0.748)	0.6860 (0.732)	0.648 (0.702)	0.459 (0.488)
	ISOLET J-R:		**NMI**	
1024	0.729 (0.761)	0.728 (0.758)	0.729 (0.761)	0.728 (0.747)
256	0.730 (0.759)	0.728 (0.756)	0.729 (0.757)	0.745 (0.776)
64	0.728 (0.758)	0.730 (0.759)	0.730 (0.756)	0.746 (0.789)
32	0.729 (0.760)	0.729 (0.759)	0.728 (0.759)	0.754 (0.804)
16	0.728 (0.762)	0.729 (0.759)	0.731 (0.744)	0.782 (0.838)
8	0.731 (0.760)	0.730 (0.747)	0.731 (0.757)	0.797 (0.844)
4	0.732 (0.748)	0.729 (0.756)	0.731 (0.765)	0.789 (0.859)
2	0.731 (0.762)	0.732 (0.758)	0.737 (0.770)	0.800 (0.875)
1	0.738 (0.772)	0.738 (0.777)	0.740 (0.772)	0.789 (0.866)
0.5	0.760 (0.788)	0.756 (0.790)	0.745 (0.786)	0.832 (0.883)
0.25	0.757 (0.800)	0.760 (0.792)	0.723 (0.753)	0.514 (0.540)
	ISOLET S-Z:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.732 (0.795)	0.736 (0.793)	0.730 (0.793)	0.732 (0.785)
256	0.738 (0.794)	0.734 (0.794)	0.727 (0.794)	0.705 (0.749)
64	0.732 (0.794)	0.720 (0.794)	0.737 (0.794)	0.681 (0.747)
32	0.733 (0.795)	0.730 (0.794)	0.733 (0.794)	0.687 (0.783)
16	0.730 (0.794)	0.735 (0.794)	0.742 (0.794)	0.761 (0.881)
8	0.725 (0.793)	0.731 (0.794)	0.730 (0.794)	0.773 (0.897)
4	0.740 (0.795)	0.724 (0.793)	0.727 (0.791)	0.764 (0.933)
2	0.754 (0.804)	0.733 (0.798)	0.718 (0.792)	0.651 (0.743)
1	0.760 (0.808)	0.737 (0.797)	0.711 (0.773)	0.657 (0.765)
0.5	0.754 (0.828)	0.739 (0.798)	0.679 (0.766)	0.615 (0.763)
0.25	0.787 (0.846)	0.714 (0.870)	0.695 (0.751)	0.420 (0.432)
	ISOLET S-Z:		**NMI**	
1024	0.762 (0.788)	0.760 (0.788)	0.755 (0.784)	0.754 (0.782)
256	0.759 (0.788)	0.761 (0.788)	0.758 (0.783)	0.763 (0.784)
64	0.761 (0.783)	0.759 (0.789)	0.764 (0.783)	0.788 (0.845)
32	0.757 (0.790)	0.759 (0.783)	0.759 (0.786)	0.793 (0.848)
16	0.760 (0.786)	0.759 (0.783)	0.764 (0.784)	0.839 (0.897)
8	0.758 (0.786)	0.758 (0.785)	0.761 (0.787)	0.843 (0.897)
4	0.765 (0.792)	0.762 (0.792)	0.763 (0.789)	0.822 (0.893)
2	0.777 (0.806)	0.766 (0.802)	0.758 (0.802)	0.822 (0.870)
1	0.777 (0.812)	0.778 (0.815)	0.759 (0.806)	0.815 (0.869)
0.5	0.780 (0.817)	0.768 (0.796)	0.758 (0.799)	0.811 (0.861)
0.25	0.803 (0.843)	0.788 (0.878)	0.781 (0.797)	0.562 (0.585)
	LETTER A-E:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.512 (0.589)	0.513 (0.589)	0.512 (0.589)	0.513 (0.589)
256	0.512 (0.589)	0.513 (0.589)	0.511 (0.588)	0.503 (0.504)
64	0.510 (0.589)	0.514 (0.589)	0.510 (0.589)	0.487 (0.488)
32	0.512 (0.589)	0.516 (0.589)	0.513 (0.589)	0.465 (0.509)
16	0.512 (0.588)	0.511 (0.588)	0.512 (0.589)	0.476 (0.517)
8	0.508 (0.588)	0.512 (0.588)	0.511 (0.588)	0.463 (0.491)
4	0.516 (0.586)	0.510 (0.586)	0.512 (0.587)	0.482 (0.490)
2	0.512 (0.586)	0.513 (0.586)	0.509 (0.585)	0.477 (0.489)
1	0.514 (0.582)	0.513 (0.583)	0.511 (0.583)	0.508 (0.528)
0.5	0.516 (0.567)	0.513 (0.578)	0.505 (0.577)	0.539 (0.585)
0.25	0.516 (0.520)	0.502 (0.562)	0.497 (0.501)	0.520 (0.595)
	LETTER A-E:		**NMI**	
1024	0.346 (0.422)	0.347 (0.422)	0.346 (0.422)	0.347 (0.422)
256	0.345 (0.422)	0.347 (0.422)	0.344 (0.422)	0.313 (0.352)
64	0.344 (0.422)	0.348 (0.422)	0.344 (0.421)	0.328 (0.329)
32	0.346 (0.422)	0.350 (0.421)	0.347 (0.421)	0.328 (0.395)
16	0.346 (0.421)	0.345 (0.421)	0.346 (0.421)	0.338 (0.409)
8	0.343 (0.421)	0.346 (0.420)	0.345 (0.420)	0.333 (0.363)
4	0.348 (0.419)	0.344 (0.420)	0.345 (0.420)	0.352 (0.362)
2	0.343 (0.419)	0.344 (0.420)	0.343 (0.420)	0.349 (0.390)
1	0.345 (0.417)	0.344 (0.418)	0.344 (0.418)	0.383 (0.410)
0.5	0.344 (0.414)	0.345 (0.418)	0.342 (0.416)	0.397 (0.468)
0.25	0.340 (0.348)	0.328 (0.410)	0.328 (0.331)	0.364 (0.458)

	LETTER F-J:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.489 (0.545)	0.486 (0.544)	0.487 (0.546)	0.489 (0.544)
256	0.489 (0.544)	0.486 (0.544)	0.487 (0.544)	0.460 (0.511)
64	0.488 (0.545)	0.488 (0.584)	0.488 (0.543)	0.501 (0.564)
32	0.488 (0.544)	0.489 (0.540)	0.491 (0.543)	0.538 (0.595)
16	0.487 (0.546)	0.489 (0.546)	0.486 (0.543)	0.599 (0.622)
8	0.489 (0.584)	0.486 (0.543)	0.489 (0.543)	0.619 (0.647)
4	0.486 (0.532)	0.489 (0.543)	0.488 (0.541)	0.597 (0.632)
2	0.486 (0.543)	0.486 (0.541)	0.489 (0.540)	0.571 (0.628)
1	0.489 (0.542)	0.492 (0.540)	0.490 (0.540)	0.589 (0.649)
0.5	0.487 (0.534)	0.489 (0.539)	0.491 (0.540)	0.470 (0.586)
0.25	0.475 (0.507)	0.492 (0.539)	0.495 (0.543)	0.315 (0.393)
	LETTER F-J:		**NMI**	
1024	0.337 (0.389)	0.333 (0.406)	0.333 (0.389)	0.334 (0.389)
256	0.336 (0.406)	0.332 (0.389)	0.336 (0.405)	0.286 (0.359)
64	0.333 (0.385)	0.335 (0.412)	0.332 (0.385)	0.377 (0.393)
32	0.334 (0.406)	0.335 (0.404)	0.337 (0.385)	0.405 (0.453)
16	0.334 (0.389)	0.335 (0.389)	0.331 (0.384)	0.437 (0.463)
8	0.336 (0.412)	0.333 (0.384)	0.335 (0.406)	0.469 (0.517)
4	0.334 (0.383)	0.334 (0.404)	0.335 (0.383)	0.444 (0.505)
2	0.328 (0.384)	0.334 (0.404)	0.339 (0.384)	0.429 (0.506)
1	0.328 (0.384)	0.337 (0.384)	0.341 (0.385)	0.445 (0.524)
0.5	0.319 (0.373)	0.336 (0.385)	0.345 (0.386)	0.350 (0.455)
0.25	0.303 (0.368)	0.342 (0.386)	0.352 (0.382)	0.153 (0.269)
	LETTER K-O:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.470 (0.481)	0.471 (0.481)	0.467 (0.481)	0.469 (0.480)
256	0.471 (0.481)	0.468 (0.480)	0.465 (0.481)	0.437 (0.437)
64	0.467 (0.481)	0.469 (0.481)	0.470 (0.480)	0.376 (0.421)
32	0.471 (0.480)	0.470 (0.481)	0.469 (0.480)	0.404 (0.502)
16	0.471 (0.481)	0.468 (0.481)	0.468 (0.481)	0.435 (0.487)
8	0.468 (0.481)	0.473 (0.481)	0.468 (0.480)	0.502 (0.556)
4	0.472 (0.481)	0.470 (0.481)	0.467 (0.479)	0.486 (0.555)
2	0.469 (0.479)	0.469 (0.478)	0.469 (0.475)	0.500 (0.560)
1	0.465 (0.476)	0.455 (0.461)	0.432 (0.455)	0.479 (0.512)
0.5	0.474 (0.474)	0.443 (0.451)	0.443 (0.510)	0.492 (0.534)
0.25	0.473 (0.487)	0.432 (0.500)	0.409 (0.455)	0.436 (0.481)
	LETTER K-O:		**NMI**	
1024	0.224 (0.238)	0.225 (0.238)	0.221 (0.238)	0.223 (0.238)
256	0.225 (0.238)	0.221 (0.237)	0.218 (0.238)	0.269 (0.271)
64	0.220 (0.238)	0.224 (0.238)	0.224 (0.238)	0.227 (0.248)
32	0.225 (0.237)	0.225 (0.238)	0.223 (0.237)	0.264 (0.331)
16	0.225 (0.238)	0.221 (0.238)	0.222 (0.238)	0.303 (0.324)
8	0.222 (0.238)	0.227 (0.238)	0.223 (0.235)	0.372 (0.406)
4	0.227 (0.239)	0.224 (0.238)	0.221 (0.233)	0.366 (0.413)
2	0.224 (0.237)	0.222 (0.234)	0.224 (0.229)	0.379 (0.422)
1	0.217 (0.232)	0.215 (0.226)	0.208 (0.242)	0.344 (0.376)
0.5	0.234 (0.234)	0.239 (0.250)	0.254 (0.298)	0.347 (0.430)
0.25	0.260 (0.288)	0.262 (0.301)	0.223 (0.270)	0.319 (0.390)
	LETTER P-T:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.554 (0.554)	0.552 (0.554)	0.554 (0.554)	0.554 (0.555)
256	0.554 (0.554)	0.554 (0.554)	0.554 (0.554)	0.509 (0.512)
64	0.554 (0.554)	0.554 (0.555)	0.554 (0.554)	0.457 (0.483)
32	0.553 (0.604)	0.554 (0.555)	0.554 (0.555)	0.445 (0.463)
16	0.554 (0.554)	0.552 (0.555)	0.554 (0.555)	0.434 (0.474)
8	0.552 (0.554)	0.554 (0.555)	0.554 (0.554)	0.467 (0.513)
4	0.554 (0.554)	0.554 (0.554)	0.555 (0.555)	0.530 (0.612)
2	0.553 (0.553)	0.553 (0.553)	0.553 (0.556)	0.550 (0.621)
1	0.553 (0.555)	0.552 (0.556)	0.551 (0.555)	0.525 (0.583)
0.5	0.550 (0.551)	0.548 (0.553)	0.535 (0.543)	0.524 (0.619)
0.25	0.542 (0.542)	0.524 (0.532)	0.498 (0.502)	0.499 (0.543)
	LETTER P-T:		**NMI**	
1024	0.371 (0.372)	0.370 (0.372)	0.371 (0.372)	0.372 (0.373)
256	0.371 (0.372)	0.371 (0.372)	0.372 (0.373)	0.305 (0.307)
64	0.371 (0.372)	0.372 (0.373)	0.372 (0.373)	0.283 (0.367)
32	0.370 (0.422)	0.372 (0.373)	0.372 (0.373)	0.285 (0.301)
16	0.371 (0.372)	0.370 (0.373)	0.372 (0.373)	0.289 (0.349)
8	0.369 (0.372)	0.372 (0.373)	0.372 (0.372)	0.318 (0.380)
4	0.370 (0.371)	0.372 (0.372)	0.372 (0.372)	0.402 (0.479)
2	0.370 (0.370)	0.372 (0.372)	0.373 (0.377)	0.417 (0.499)
1	0.368 (0.370)	0.370 (0.375)	0.371 (0.374)	0.397 (0.449)
0.5	0.364 (0.365)	0.364 (0.372)	0.351 (0.357)	0.396 (0.455)
0.25	0.348 (0.348)	0.332 (0.339)	0.309 (0.311)	0.372 (0.438)

	LETTER U-Z:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.511 (0.558)	0.517 (0.558)	0.505 (0.558)	0.505 (0.558)
256	0.508 (0.558)	0.515 (0.558)	0.507 (0.558)	0.479 (0.493)
64	0.509 (0.558)	0.503 (0.558)	0.503 (0.558)	0.480 (0.522)
32	0.510 (0.558)	0.502 (0.558)	0.500 (0.558)	0.493 (0.539)
16	0.507 (0.558)	0.504 (0.558)	0.509 (0.557)	0.496 (0.514)
8	0.508 (0.559)	0.504 (0.558)	0.509 (0.557)	0.481 (0.524)
4	0.509 (0.558)	0.513 (0.558)	0.512 (0.557)	0.486 (0.527)
2	0.506 (0.560)	0.496 (0.557)	0.500 (0.557)	0.495 (0.560)
1	0.509 (0.559)	0.495 (0.557)	0.501 (0.555)	0.445 (0.571)
0.5	0.495 (0.562)	0.497 (0.557)	0.492 (0.550)	0.466 (0.551)
0.25	0.498 (0.567)	0.478 (0.550)	0.486 (0.533)	0.481 (0.585)
	LETTER U-Z:		**NMI**	
1024	0.395 (0.437)	0.403 (0.437)	0.394 (0.437)	0.390 (0.437)
256	0.395 (0.437)	0.402 (0.436)	0.395 (0.437)	0.368 (0.394)
64	0.397 (0.436)	0.389 (0.436)	0.390 (0.436)	0.354 (0.404)
32	0.396 (0.436)	0.391 (0.437)	0.388 (0.435)	0.367 (0.422)
16	0.392 (0.436)	0.391 (0.436)	0.395 (0.437)	0.392 (0.411)
8	0.396 (0.437)	0.391 (0.436)	0.397 (0.436)	0.400 (0.459)
4	0.395 (0.437)	0.401 (0.436)	0.399 (0.436)	0.423 (0.473)
2	0.395 (0.437)	0.386 (0.435)	0.385 (0.435)	0.437 (0.495)
1	0.396 (0.438)	0.383 (0.435)	0.387 (0.432)	0.425 (0.502)
0.5	0.382 (0.436)	0.383 (0.431)	0.378 (0.426)	0.425 (0.471)
0.25	0.389 (0.445)	0.371 (0.428)	0.377 (0.411)	0.426 (0.481)
	NEWS20:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.235 (0.248)	0.234 (0.248)	0.236 (0.248)	0.235 (0.248)
256	0.236 (0.250)	0.235 (0.250)	0.235 (0.248)	0.236 (0.251)
64	0.235 (0.246)	0.235 (0.247)	0.236 (0.250)	0.281 (0.307)
32	0.236 (0.249)	0.235 (0.248)	0.236 (0.248)	0.353 (0.386)
16	0.236 (0.251)	0.236 (0.250)	0.236 (0.246)	0.378 (0.419)
8	0.236 (0.251)	0.236 (0.250)	0.236 (0.257)	0.355 (0.382)
4	0.238 (0.252)	0.237 (0.248)	0.238 (0.250)	0.238 (0.254)
2	0.242 (0.256)	0.242 (0.262)	0.241 (0.254)	0.191 (0.206)
1	0.244 (0.264)	0.245 (0.261)	0.244 (0.259)	0.064 (0.068)
0.5	0.198 (0.212)	0.191 (0.205)	0.190 (0.205)	0.090 (0.095)
0.25	0.131 (0.148)	0.135 (0.142)	0.133 (0.146)	0.082 (0.086)
	NEWS20:		**NMI**	
1024	0.223 (0.238)	0.223 (0.244)	0.224 (0.237)	0.223 (0.238)
256	0.224 (0.236)	0.222 (0.238)	0.222 (0.239)	0.224 (0.244)
64	0.223 (0.236)	0.222 (0.238)	0.223 (0.238)	0.289 (0.310)
32	0.223 (0.241)	0.222 (0.238)	0.223 (0.241)	0.371 (0.394)
16	0.224 (0.241)	0.222 (0.240)	0.225 (0.240)	0.416 (0.430)
8	0.224 (0.239)	0.225 (0.239)	0.224 (0.243)	0.422 (0.433)
4	0.226 (0.240)	0.225 (0.246)	0.225 (0.239)	0.345 (0.362)
2	0.229 (0.242)	0.229 (0.249)	0.229 (0.242)	0.217 (0.224)
1	0.241 (0.254)	0.241 (0.254)	0.238 (0.252)	0.061 (0.066)
0.5	0.219 (0.228)	0.217 (0.226)	0.217 (0.227)	0.029 (0.033)
0.25	0.119 (0.130)	0.109 (0.122)	0.114 (0.127)	0.025 (0.027)
	OPTDIGIT:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.759 (0.801)	0.750 (0.801)	0.751 (0.801)	0.766 (0.798)
256	0.763 (0.801)	0.758 (0.798)	0.760 (0.801)	0.733 (0.849)
64	0.756 (0.795)	0.755 (0.801)	0.755 (0.797)	0.790 (0.840)
32	0.763 (0.801)	0.759 (0.798)	0.761 (0.801)	0.797 (0.857)
16	0.764 (0.801)	0.760 (0.801)	0.746 (0.798)	0.798 (0.871)
8	0.751 (0.802)	0.757 (0.797)	0.759 (0.798)	0.816 (0.878)
4	0.754 (0.802)	0.759 (0.798)	0.753 (0.797)	0.829 (0.880)
2	0.769 (0.804)	0.754 (0.800)	0.754 (0.794)	0.848 (0.911)
1	0.764 (0.803)	0.761 (0.800)	0.762 (0.793)	0.807 (0.903)
0.5	0.791 (0.803)	0.767 (0.795)	0.747 (0.793)	0.658 (0.831)
0.25	0.786 (0.814)	0.761 (0.824)	0.704 (0.767)	0.624 (0.708)
	OPTDIGIT:		**NMI**	
1024	0.698 (0.713)	0.694 (0.713)	0.693 (0.713)	0.707 (0.729)
256	0.699 (0.729)	0.697 (0.711)	0.697 (0.713)	0.713 (0.759)
64	0.696 (0.712)	0.696 (0.714)	0.695 (0.711)	0.768 (0.798)
32	0.699 (0.713)	0.698 (0.728)	0.698 (0.728)	0.796 (0.823)
16	0.699 (0.713)	0.698 (0.730)	0.693 (0.729)	0.811 (0.839)
8	0.694 (0.715)	0.696 (0.717)	0.697 (0.715)	0.830 (0.850)
4	0.696 (0.719)	0.698 (0.731)	0.695 (0.717)	0.845 (0.868)
2	0.704 (0.723)	0.697 (0.715)	0.697 (0.722)	0.874 (0.897)
1	0.707 (0.733)	0.705 (0.744)	0.705 (0.741)	0.862 (0.890)
0.5	0.728 (0.758)	0.718 (0.748)	0.709 (0.750)	0.782 (0.843)
0.25	0.748 (0.758)	0.725 (0.755)	0.694 (0.722)	0.655 (0.672)

	PENDIGIT:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.705 (0.792)	0.703 (0.793)	0.706 (0.792)	0.729 (0.785)
256	0.703 (0.795)	0.698 (0.793)	0.694 (0.790)	0.710 (0.788)
64	0.691 (0.791)	0.702 (0.794)	0.707 (0.787)	0.731 (0.830)
32	0.706 (0.790)	0.702 (0.772)	0.707 (0.790)	0.736 (0.833)
16	0.701 (0.789)	0.695 (0.791)	0.701 (0.796)	0.738 (0.837)
8	0.707 (0.771)	0.702 (0.785)	0.708 (0.784)	0.738 (0.841)
4	0.704 (0.783)	0.697 (0.784)	0.710 (0.785)	0.756 (0.847)
2	0.716 (0.786)	0.721 (0.785)	0.712 (0.790)	0.737 (0.857)
1	0.725 (0.768)	0.729 (0.774)	0.730 (0.783)	0.690 (0.793)
0.5	0.730 (0.781)	0.732 (0.787)	0.722 (0.786)	0.566 (0.703)
0.25	0.724 (0.764)	0.726 (0.793)	0.733 (0.820)	0.469 (0.523)
	PENDIGIT:		**NMI**	
1024	0.681 (0.707)	0.679 (0.719)	0.682 (0.707)	0.687 (0.713)
256	0.679 (0.716)	0.678 (0.716)	0.677 (0.709)	0.678 (0.710)
64	0.677 (0.704)	0.679 (0.718)	0.681 (0.704)	0.713 (0.753)
32	0.681 (0.710)	0.680 (0.717)	0.680 (0.710)	0.734 (0.772)
16	0.679 (0.709)	0.677 (0.717)	0.680 (0.717)	0.744 (0.782)
8	0.680 (0.717)	0.678 (0.709)	0.680 (0.708)	0.753 (0.798)
4	0.672 (0.710)	0.671 (0.718)	0.675 (0.709)	0.776 (0.804)
2	0.670 (0.695)	0.672 (0.704)	0.670 (0.703)	0.775 (0.814)
1	0.674 (0.698)	0.679 (0.690)	0.678 (0.697)	0.774 (0.826)
0.5	0.693 (0.710)	0.693 (0.707)	0.689 (0.708)	0.666 (0.730)
0.25	0.679 (0.706)	0.688 (0.709)	0.705 (0.734)	0.491 (0.508)
	SATIMAGE:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.514 (0.535)	0.514 (0.534)	0.514 (0.562)	0.516 (0.535)
256	0.514 (0.535)	0.515 (0.562)	0.513 (0.538)	0.523 (0.545)
64	0.515 (0.535)	0.512 (0.535)	0.513 (0.534)	0.562 (0.584)
32	0.514 (0.562)	0.513 (0.562)	0.515 (0.535)	0.569 (0.609)
16	0.515 (0.535)	0.514 (0.535)	0.512 (0.535)	0.578 (0.616)
8	0.514 (0.534)	0.514 (0.534)	0.517 (0.561)	0.580 (0.614)
4	0.515 (0.535)	0.510 (0.559)	0.515 (0.533)	0.601 (0.623)
2	0.514 (0.559)	0.518 (0.534)	0.512 (0.558)	0.608 (0.625)
1	0.514 (0.558)	0.513 (0.532)	0.519 (0.531)	0.617 (0.628)
0.5	0.521 (0.531)	0.521 (0.533)	0.522 (0.533)	0.611 (0.639)
0.25	0.565 (0.582)	0.569 (0.588)	0.573 (0.590)	0.609 (0.639)
	SATIMAGE:		**NMI**	
1024	0.404 (0.422)	0.403 (0.425)	0.404 (0.460)	0.405 (0.425)
256	0.406 (0.425)	0.405 (0.460)	0.404 (0.425)	0.406 (0.431)
64	0.404 (0.424)	0.405 (0.425)	0.406 (0.425)	0.493 (0.511)
32	0.405 (0.460)	0.406 (0.460)	0.404 (0.424)	0.492 (0.518)
16	0.406 (0.425)	0.406 (0.424)	0.402 (0.423)	0.503 (0.531)
8	0.402 (0.425)	0.405 (0.426)	0.405 (0.459)	0.505 (0.531)
4	0.404 (0.425)	0.406 (0.459)	0.405 (0.427)	0.511 (0.568)
2	0.404 (0.460)	0.405 (0.423)	0.402 (0.460)	0.591 (0.612)
1	0.406 (0.459)	0.405 (0.429)	0.405 (0.424)	0.600 (0.618)
0.5	0.406 (0.429)	0.408 (0.428)	0.400 (0.433)	0.598 (0.623)
0.25	0.473 (0.483)	0.491 (0.511)	0.494 (0.511)	0.603 (0.622)
	SHUTTLE:		**Accuracy**	
γ	RA	Ncut	SK	BBS
1024	0.369 (0.457)	0.372 (0.452)	0.368 (0.453)	0.371 (0.428)
256	0.371 (0.452)	0.365 (0.452)	0.368 (0.452)	0.363 (0.465)
64	0.365 (0.464)	0.375 (0.469)	0.364 (0.444)	0.376 (0.479)
32	0.384 (0.463)	0.376 (0.463)	0.376 (0.462)	0.398 (0.521)
16	0.336 (0.431)	0.338 (0.421)	0.339 (0.427)	0.406 (0.553)
8	0.331 (0.397)	0.333 (0.423)	0.338 (0.422)	0.396 (0.483)
4	0.331 (0.420)	0.335 (0.425)	0.338 (0.421)	0.565 (0.679)
2	0.329 (0.433)	0.340 (0.429)	0.346 (0.397)	0.567 (0.861)
1	0.350 (0.474)	0.356 (0.414)	0.362 (0.420)	0.531 (0.679)
0.5	0.350 (0.423)	0.366 (0.424)	0.453 (0.506)	0.647 (0.795)
0.25	0.349 (0.416)	0.448 (0.506)	0.423 (0.510)	0.330 (0.362)
	SHUTTLE:		**NMI**	
1024	0.385 (0.474)	0.384 (0.474)	0.384 (0.481)	0.389 (0.463)
256	0.388 (0.481)	0.383 (0.474)	0.381 (0.469)	0.417 (0.492)
64	0.382 (0.471)	0.393 (0.472)	0.383 (0.489)	0.440 (0.559)
32	0.387 (0.484)	0.376 (0.469)	0.379 (0.469)	0.469 (0.563)
16	0.330 (0.507)	0.338 (0.484)	0.339 (0.443)	0.473 (0.561)
8	0.326 (0.444)	0.335 (0.450)	0.336 (0.437)	0.468 (0.563)
4	0.336 (0.441)	0.342 (0.451)	0.345 (0.451)	0.514 (0.595)
2	0.342 (0.445)	0.361 (0.477)	0.366 (0.453)	0.542 (0.705)
1	0.391 (0.473)	0.404 (0.507)	0.403 (0.480)	0.386 (0.452)
0.5	0.396 (0.506)	0.417 (0.482)	0.448 (0.483)	0.206 (0.215)
0.25	0.392 (0.516)	0.429 (0.490)	0.392 (0.404)	0.220 (0.272)