Submitted 1/12; Revised 11/12; Published 7/13

Similarity-based Clustering by Left-Stochastic Matrix Factorization

Raman Arora ARORA@TTIC.EDU
Toyota Technological Institute
6045 S. Kenwood Ave
Chicago, IL 60637, USA

Maya R. Gupta MAYAGUPTA@GOOGLE.COM
Google
1225 Charleston Rd
Mountain View, CA 94301, USA

Amol Kapila AKAPILA@U.WASHINGTON.EDU
Maryam Fazel MFAZEL@U.WASHINGTON.EDU
Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA

Editor: Inderjit Dhillon

Abstract

For similarity-based clustering, we propose modeling the entries of a given similarity matrix as the inner products of the unknown cluster probabilities. To estimate the cluster probabilities from the given similarity matrix, we introduce a left-stochastic non-negative matrix factorization problem. A rotation-based algorithm is proposed for the matrix factorization. Conditions for unique matrix factorizations and clusterings are given, and an error bound is provided. The algorithm is particularly efficient for the case of two clusters, which motivates a hierarchical variant for cases where the number of desired clusters is large. Experiments show that the proposed left-stochastic decomposition clustering model produces relatively high within-cluster similarity on most data sets and can match given class labels, and that the efficient hierarchical variant performs surprisingly well.

Keywords: clustering, non-negative matrix factorization, rotation, indefinite kernel, similarity, completely positive

1. Introduction

Clustering is important in a broad range of applications, from segmenting customers for more effective advertising, to building codebooks for data compression. Many clustering methods can be interpreted in terms of a matrix factorization problem. For example, the popular k-means clustering algorithm attempts to solve the *k-means problem*: produce a clustering such that the sum of squared error between samples and the mean of their cluster is small (Hastie et al., 2009). For n feature vectors gathered as the d-dimensional columns of a matrix $X \in \mathbb{R}^{d\times n}$, the k-means problem can be written as a matrix factorization:

$$\begin{aligned} \underset{\substack{F\in\mathbb{R}^{d\times k}\\ G\in\mathbb{R}^{k\times n}}}{\text{minimize}} \quad & \|X - FG\|_F^2 \\ \text{subject to} \quad & G \in \{0,1\}^{k\times n}, G^T\mathbf{1}_k = \mathbf{1}_n, \end{aligned} \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbf{1}_n$ is an $n \times 1$ vector of 1's. The matrix F can be interpreted as a matrix whose columns are the k cluster centroids. The combined constraints $G \in \{0,1\}^{k\times n}$ and $G^T\mathbf{1}_k = \mathbf{1}_n$ force each column of G to contain all zeros except for one element, which is a 1, whose location corresponds to the cluster assignment. That is, $G_{ij} = 1$ if sample j belongs in cluster i, and $G_{ij} = 0$ otherwise. The k-means clustering problem is not convex. The k-means algorithm is traditionally used to seek a local optimum to (1) (Hastie et al., 2009; Selim and Ismail, 1984).

The *soft* k-means problem instead solves for a *cluster probability matrix* P that specifies the probability of each sample belonging to each of the clusters:

$$\begin{aligned} &\underset{\substack{F\in\mathbb{R}^{d\times k}\\ P\in\mathbb{R}^{k\times n}}}{\text{minimize}} && \|X - FP\|_F^2 \\ &\text{subject to} && P \geq 0, P^T\mathbf{1}_k = \mathbf{1}_n, \end{aligned} \tag{2}$$

where the inequality $P \geq 0$ is entrywise. The constraints $P \geq 0$ and $P^T\mathbf{1}_k = \mathbf{1}_n$ together force each column of P to be non-negative and to sum to 1, making P left-stochastic. Hence, each column of P is a probability mass function encoding probabilistic assignments of points to clusters: we interpret P_{ij} to express the probability that sample j belongs in cluster i.

Interpreting non-negative factors of matrices as describing a data clustering was proposed by Paatero and Tapper (1994) and Lee and Seung (1999). Following their work, other researchers have posed different matrix factorization models, and attempted to explicitly solve the resulting matrix factorization problems to form clusterings. Zha et al. (2001) proposed relaxing the constraints on G in the k-means optimization problem to an orthogonality constraint:

$$\begin{aligned} &\underset{\substack{F\in\mathbb{R}^{d\times k}\\ G\in\mathbb{R}^{k\times n}}}{\text{minimize}} && \|X - FG\|_F^2 \\ &\text{subject to} && GG^T = I_k, G \geq 0, \end{aligned}$$

where I_k is a $k \times k$ identity matrix. Ding et al. (2005) considered the kernelized clustering objective,

$$\begin{aligned} &\underset{G\in\mathbb{R}^{k\times n}}{\text{minimize}} && \|K - G^TG\|_F^2 \\ &\text{subject to} && GG^T = I_k, G \geq 0. \end{aligned}$$

Recently, Ding et al. (2010) considered changing the constraints on G in the k-means problem given in (1) to only require that G be positive. They explored a number of approximations to the k-means problem that imposed different constraints on F. One such variant that they deemed particularly worthy of further investigation was *convex NMF*. Convex NMF restricts the columns of F (the cluster centroids) to be convex combinations of the columns of X:

$$\underset{\substack{W\in\mathbb{R}_+^{n\times k}\\ G\in\mathbb{R}_+^{k\times n}}}{\text{minimize}} \quad \|X - XWG\|_F^2.$$

Convex NMF is amenable to the kernel trick (Ding et al., 2010); for an input kernel matrix K, the kernel convex NMF solves

$$\underset{\substack{W\in\mathbb{R}_+^{n\times k}\\ G\in\mathbb{R}_+^{k\times n}}}{\text{minimize}} \quad \operatorname{tr}\left(K - 2G^TW + W^TKWG^TG\right).$$

In this paper, we propose a new non-negative matrix factorization (NNMF) model for clustering from pairwise similarities between the samples.[1] First, we introduce the proposed *left-stochastic decomposition* (LSD) in Section 1.1. Then in Section 2, we provide a theoretical foundation for LSD and motivate our rotation-based algorithm, which is given in Section 3. For $k > 2$ clusters, we show that there may be multiple matrix factorization solutions related by a rotation, and provide conditions for the uniqueness of the resulting clustering. For $k = 2$ clusters, our approach to solving the LSD problem provides a simple unique solution, which motivates a fast binary hierarchical LSD clustering, described in Section 3.3. Experimental results are presented in Section 5, and show that the LSD clustering model performs well in terms of both within-class similarity and misclassification rate. The paper concludes with a discussion and some notes on how the proposed rotation-based algorithmic approach may be useful in other contexts.

1.1 Left-Stochastic Matrix Decomposition

Cristianini et al. (2001) defined the *ideal kernel* K^* to have $K^*_{ij} = 1$ if and only if the i^{th} and j^{th} sample are from the same class or cluster. We propose a clustering model that relaxes this notion of an ideal kernel. Suppose $P \in \{0,1\}^{k\times n}$ is a cluster assignment matrix with the $P_{ij} = 1$ if and only if the j^{th} sample belongs to the i^{th} cluster. Then $K^* = P^T P$ would be such an ideal kernel. Relax the ideal kernel's assumption that each sample belongs to only one class, and let $P \in [0,1]^{k\times n}$ be interpreted as a soft cluster assignment matrix for n samples and k clusters. We further constrain P to be left-stochastic, so that each column sums to one, allowing P_{ij} to be interpreted as the probability that sample j belongs to cluster i. In practice, the range of the entries of a given matrix K may preclude it from being well approximated by $P^T P$ for a left-stochastic matrix P. For example, if the elements of K are very large, then it will be impossible to approximate them by the inner products of probability vectors. For this reason, we also include a scaling factor $c \in \mathbb{R}$, producing what we term the *left-stochastic decomposition* (LSD) model: $cK \approx P^T P$.

We use the LSD model for clustering by solving a non-negative matrix factorization problem for the best cluster assignment matrix P. That is, given a matrix K, we propose finding a scaling factor c and a cluster probability matrix P that best solve

$$\begin{aligned} &\underset{\substack{c\in\mathbb{R}_+ \\ P\in[0,1]^{k\times n}}}{\text{minimize}} \quad \|K - \frac{1}{c}P^T P\|_F^2 \\ &\text{subject to} \quad P^T \mathbf{1}_k = \mathbf{1}_n. \end{aligned} \tag{3}$$

LSD Clustering: Given a solution $\hat{P}$ to (3) an LSD clustering is an assignment of samples to clusters such that if the j^{th} sample is assigned to cluster i^*, then $\hat{P}_{i^*j} \geq \hat{P}_{ij}$ for all i.

Input K: While the LSD problem stated in (3) does not in itself require any conditions on the input matrix K, we restrict our attention to symmetric input matrices whose top k eigenvalues are positive, and refer to such matrices in this paper as *similarity matrices*. If a user's input matrix does not have k positive eigenvalues, its spectrum can be easily modified (Chen et al., 2009a,b).

LSDable: K is *LSDable* if the minimum of (3) is zero, that is $cK = P^T P$ for some feasible c and P.

[1] A preliminary version of this paper appeared as Arora et al. (2011). This paper includes new theoretical results, detailed proofs, new algorithmic contributions, and more exhaustive experimental comparisons.

1.2 Equivalence of LSD Clustering and Soft k-Means

A key question for LSD clustering is, "How does it relate to the soft k-means problem?" We can show that for LSDable K, any solution to the LSD problem also solves the soft k-means problem (2):

Proposition 1 (Soft k-means Equivalence) *Suppose K is LSDable. Consider any feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that $K_{ij} = \phi(X_i)^T \phi(X_j)$, and let $\Phi(X) = [\phi(X_1), \phi(X_2), \dots, \phi(X_n)] \in \mathbb{R}^{d' \times n}$. Then the minimizer P^* of the LSD problem (3) also minimizes the following soft k-means problem:*

$$\begin{aligned} &\underset{\substack{F\in\mathbb{R}^{d'\times k}\\ P\geq 0,}}{\textit{minimize}} && \|\Phi(X) - FP\|_F^2 \\ &\textit{subject to} && P^T \mathbf{1}_k = \mathbf{1}_n. \end{aligned} \tag{4}$$

In practice, a given similarity matrix K will not be LSDable, and it is an open question how the LSD solution and soft k-means solution will be related.

2. Theory

In this section we provide a theoretical foundation for LSD clustering and our rotation-based LSD algorithm. All of the proofs are in the appendix.

2.1 Decomposition of the LSD Objective Error

To analyze LSD and motivate our algorithm, we consider the error introduced by the various constraints on P separately. Let $M \in \mathbb{R}^{k \times n}$ be any rank-k matrix (this implies $M^T M$ has rank k). Let $Z \in \mathbb{R}^{k \times n}$ be any matrix such that $Z^T Z$ is LSDable. The LSD objective given in (3) can be re-written as:

$$\begin{aligned} \|K - \frac{1}{c}P^T P\|_F^2 &= \|K - M^T M + M^T M - Z^T Z + Z^T Z - \frac{1}{c}P^T P\|_F^2 \\ &= \|K - M^T M\|_F^2 + \|M^T M - Z^T Z\|_F^2 + \|Z^T Z - \frac{1}{c}P^T P\|_F^2 && (5) \\ &\quad + \operatorname{trace}\left(2(K - M^T M)^T (Z^T Z - \frac{1}{c}P^T P)\right) && (6) \\ &\quad + \operatorname{trace}\left(2(K - M^T M)^T (M^T M - Z^T Z)\right) && (7) \\ &\quad + \operatorname{trace}\left(2(M^T M - Z^T Z)(Z^T Z - \frac{1}{c}P^T P)\right), && (8) \end{aligned}$$

where the second equality follows from the definition of the Frobenius norm. Thus minimizing the LSD objective requires minimizing the sum of the above six terms.

First, consider the third term on line (5): $\|Z^T Z - \frac{1}{c}P^T P\|_F^2$ for some LSDable $Z^T Z$. For any LSDable $Z^T Z$, by definition, we can choose a c and a P such that $Z^T Z = \frac{1}{c}P^T P$. Such a choice will zero-out the third, fourth and sixth terms on lines (5), (6) and (8) respectively. Proposition 2 gives a closed-form solution for such an optimal c, and Theorem 3 states that given such a c, the optimal matrix P is a rotation of the matrix $\sqrt{c}Z$.

The optimal choice of a rank k matrix $M^T M$ depends on the first and second terms on line (5), and the fifth term on line (7). As an approximation, in our algorithm we will choose M to minimize only the first term, $\|K - M^T M\|_F^2$. Given the eigendecomposition $K = V\Lambda V^T = \sum_{i=1}^n \lambda_i v_i v_i^T$ such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ (with $\lambda_k > 0$ by our assumption that K is a similarity matrix), the best rank-k matrix approximation in Frobenius norm (or in general any unitarily invariant norm) to matrix K is given as $K^{(k)} = \sum_{i=1}^k \lambda_i v_i v_i^T$ (Eckart and Young, 1936). Setting $M = \left[\sqrt{\lambda_1} v_1 \; \sqrt{\lambda_2} v_2 \; \cdots \sqrt{\lambda_k} v_k\right]$ will thus minimize the first term. If $K^{(k)}$ is LSDable, then setting $Z = M$ will zero-out the second and fifth terms, and this strategy will be optimal, achieving the lowerbound error $\|K - K^{(k)}\|_F^2$ on the LSD objective.

If $K^{(k)}$ is not LSDable, then it remains to choose Z. The remaining error would be minimized by choosing the Z that minimizes the second and fifth terms, subject to $Z^T Z$ being LSDable:

$$\|K^{(k)} - Z^T Z\|_F^2 + \text{trace}\left(2(K - K^{(k)})^T (K^{(k)} - Z^T Z)\right). \tag{9}$$

As a heuristic to efficiently approximately minimizing (9), we take advantage of the fact that if $Z^T Z$ is LSDable then the columns of Z must lie on a hyperplane (because the columns of P add to one, and so they lie on a hyperplane, and by Theorem 3, Z is simply a scaled rotation of P, and thus its columns must also lie on a hyperplane). So we set the columns of Z to be the least-squares hyperplane fit to the columns of M. This in itself is not enough to make $Z^T Z$ LSDable—the convex hull of its columns must fit in the probability simplex, which is achieved by projecting the columns on to the probability simplex. In fact we learn an appropriate rotation and projection via an alternating minimization approach. See Section 3.4 for more details.

See Section 3.2 for complete details on the proposed LSD algorithm, and Section 3.1 for alternative algorithmic approaches.

2.2 Optimal Scaling Factor for LSD

The following proposition shows that given any LSDable matrix $Z^T Z$, the optimal c has a closed-form solution and is independent of the optimal P. Thus, without loss of generality, other results in this paper assume that $c^* = 1$.

Proposition 2 (Scaling Factor) *Let $Z \in \mathbb{R}^{k \times n}$ be a rank-k matrix such that $Z^T Z$ is LSDable. Then, the scaling factor c^* that solves*

$$\begin{aligned} \underset{\substack{c \in \mathbb{R}_+ \\ P \in [0,1]^{k \times n}}}{\textit{minimize}} \quad & \|Z^T Z - \frac{1}{c} P^T P\|_F^2 \\ \textit{subject to} \quad & P^T \mathbf{1}_k = \mathbf{1}_n \end{aligned}$$

is

$$c^* = \frac{\|(ZZ^T)^{-1} Z \mathbf{1}_n\|_2^2}{k}. \tag{10}$$

Further, let $Q \in \mathbb{R}^{k \times n}$ be some matrix such that $Q^T Q = Z^T Z$, then c^* can equivalently be written in terms of Q rather than Z.

Note that the matrix ZZ^T in (10) is invertible because Z is a full rank matrix.

2.3 Family of LSD Factorizations

For a given similarity matrix K, there will be multiple left-stochastic factors that are related by rotations about the normal to the probability simplex (which includes permuting the rows, that is, changing the cluster labels):

Theorem 3 (Factors of K Related by Rotation) *Suppose K is* LSDable *such that $K = P^T P$. Then,* (a) *for any matrix $M \in \mathbb{R}^{k\times n}$ s.t. $K = M^T M$, there is a unique orthogonal $k \times k$ matrix R s.t. $M = RP$.* (b) *for any matrix $\hat{P} \in \mathbb{R}^{k\times n}$ s.t. $\hat{P} \geq 0, \hat{P}^T \mathbf{1}_k = \mathbf{1}_n$ and $K = \hat{P}^T \hat{P}$, there is a unique orthogonal $k \times k$ matrix R_u s.t. $\hat{P} = R_u P$ and $R_u u = u$, where $u = \frac{1}{k}[1, \dots, 1]^T$ is normal to the plane containing the probability simplex.*

2.4 Uniqueness of an LSD Clustering

While there may be many left-stochastic decompositions of a given K, they may all result in the same LSD clustering. The theorem below gives sufficient conditions for the uniqueness of an LSD clustering. For $k = 2$ clusters, the theorem implies that the LSD clustering is unique. For $k = 3$ clusters, the theorem's condition may be restated as follows (see Figure 1 for an illustration): Let α_c (α_{ac}) be the maximum angle of clockwise (anti-clockwise) rotation about the normal to the simplex that does not rotate any of the columns of P off the simplex; let β_c (β_{ac}) be the minimum angle of clockwise (anti-clockwise) rotation that changes the clustering. Then, if $\alpha_c < \beta_c$ and $\alpha_{ac} < \beta_{ac}$, the LSD clustering is unique, up to a permutation of labels. This condition for $k = 3$ can be checked in linear time.

Theorem 4 (Unique LSD Clustering) *Let H_u be the subgroup of the rotation group $SO(k)$ in $\mathbb{R}^k$ that leaves the vector $u = \frac{1}{k}[1, \dots, 1]^T \in \mathbb{R}^k$ invariant. Consider the subset*

$$G(P) = \{R \in H_u \mid (RP)_j \in \Delta^k, j = 1, \dots, n\},$$

of rotation matrices that rotate all columns of a given LSD factor P into the probability simplex $\Delta^k \subset \mathbb{R}^k$. Consider two partitions of $G(P)$: the first partition $G(P) = \cup_i G_i^{(LSD)}(P)$ such that $R', R'' \in G_i^{(LSD)}(P)$ if and only if R', R'' result in the same LSD clustering; the second partition $G(P) = \cup_i G_i^{(conn)}(P)$ such that $R', R'' \in G_i^{(conn)}(P)$ if and only if there is a continuous connected path between R' and R'' that is contained in $G_i^{(conn)}(P)$. Let $\mu(\cdot)$ denote the Haar measure on the rotation group $SO(k-1)$ and define

$$\begin{aligned} \alpha &= \sup_{i \in I^{(conn)}} \mu(G_i^{(conn)}(P)), \\ \beta &= \inf_{i \in I^{(LSD)}} \mu(G_i^{(LSD)}(P)). \end{aligned}$$

Then, if $\alpha < \beta$, the LSD clustering is unique, up to a permutation of labels.

2.5 Error Bounds on LSD

Our next result uses perturbation theory (Stewart, 1998) to give an error bound on how much an LSD factor is perturbed when an LSDable matrix K is additively perturbed.

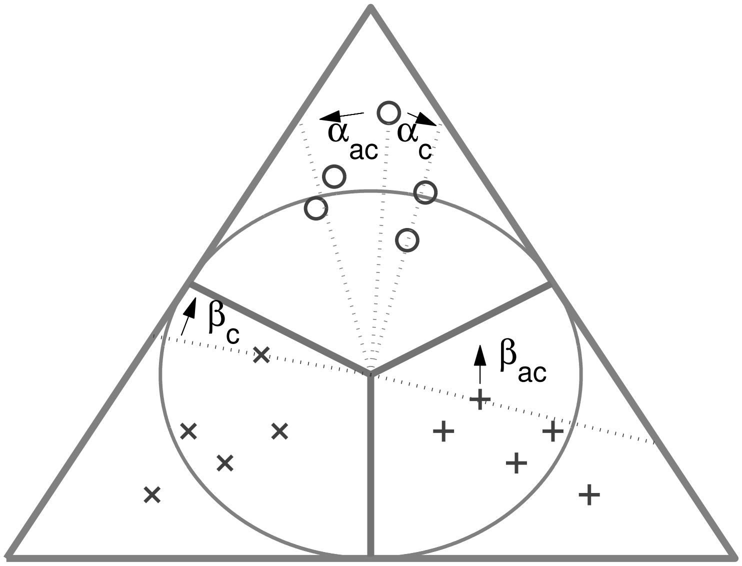


Figure 1: Illustration of conditions for uniqueness of the LSD clustering for the case $k = 3$ and for an LSDable K. The columns of P^* are shown as points on the probability simplex for $n = 15$ samples. The Y-shaped thick gray lines show the clustering decision regions, and separate the 15 points into three clusters marked by 'o's, '+'s, and 'x's. One can rotate the columns of P^* about the center of the simplex u to form a different probability matrix $\hat{P} = R_u P^*$, but the inner product does not change $\hat{P}^T \hat{P} = P^{*T} P^* = K$, thus any such rotation is another LSD solution. One can rotate clockwise by at most angle β_c (and anti-clockwise β_{ac}) before a point crosses a cluster decision boundary, which changes the clustering. Note that rotating P^* by more than α_c degrees clockwise (or α_{ac} degrees anti-clockwise) pushes the points out of the probability simplex - no longer a legitimate LSD; but rotating by 120 degrees is a legitimate LSD that corresponds to a re-labeling of the clusters. Theorem 4 applied to the special case of $k = 3$ states that a sufficient condition for the LSD clustering to be unique (up to re-labeling) is if $\alpha_c < \beta_c$ and $\alpha_{ac} < \beta_{ac}$. That is, if one cannot rotate any point across a cluster boundary without pushing another point out of the probability simplex, then the LSD clustering is unique.

Theorem 5 (Perturbation Error Bound) *Suppose K is* LSDable *and let $\tilde{K} = K + W$, where W is a symmetric matrix with bounded Frobenius norm, $\|W\|_F \le \varepsilon$. Then $\|K - \hat{P}^T \hat{P}\|_F \le 2\varepsilon$, where $\hat{P}$ minimizes (3) for $\tilde{K}$. Furthermore, if $\|W\|_F$ is $o(\tilde{\lambda}_k)$, where $\tilde{\lambda}_k$ is the k^{th} largest eigenvalue of $\tilde{K}$, then there exists an orthogonal matrix R and a constant C_1 such that*

$$\|P - R\hat{P}\|_F \le \varepsilon \left(1 + C_1 \frac{\sqrt{k}}{|\tilde{\lambda}_k|} \left(\sqrt{\mathrm{tr}(K)} + \varepsilon \right) \right). \tag{11}$$

The error bound in (11) involves three terms: the first term captures the perturbation of the eigenvalues and scales linearly with ε; the second term involves $\|K^{\frac{1}{2}}\|_F = \sqrt{\mathrm{tr}(K)}$ due to the coupling between the eigenvectors of the original matrix K and the perturbed eigenvectors as well as perturbed eigenvalues; and the third term proportional to ε^2 is due to the perturbed eigenvectors and perturbed eigenvalues. As expected, $\|P - R\hat{P}\|_F \to 0$ as $\varepsilon \to 0$, relating the LSD to the true factor with a rotation, which is consistent with Theorem 3.

3. LSD Algorithms

The LSD problem (3) is a nonconvex optimization problem in P. Standard NNMF techniques could be adapted to optimize it, as we discuss in Section 3.1. In Section 3.2, we propose a new rotation-based iterative algorithm that exploits the invariance of inner products to rotations to solve (3). For the special case of $k = 2$ clusters, the proposed algorithm requires no iterations. The simple $k = 2$ clustering can be used to produce a fast hierarchical binary-tree clustering, which we discuss in Section 3.3.

3.1 NNMF Algorithms to Solve LSD

The LSD problem stated in (3) is a *completely positive* (CP) matrix factorization with an additional left-stochastic constraint (Berman and Shaked-Monderer, 2003). CP problems are a subset of non-negative matrix factorization (NNMF) problems where a matrix A is factorized as $A = B^T B$ and B has non-negative entries. We are not aware of any NNMF algorithms specifically designed to solve problems equivalent to (3). A related NNMF problem has been studied in Ho (2008), where the matrix K to be factorized had fixed column sums, rather than our problem where the constraint is that the matrix factor P has fixed column sums.

A survey of NNMF algorithms can be found in Berry et al. (2007). Standard NNMF approaches can be used to solve the LSD problem with appropriate modifications. We adapted the multiplicative update method of Lee and Seung (2000) for the LSD problem constraints by adding a projection onto the feasible set at each iteration. In Section 5, we show experimentally that both the proposed rotation-based approach and the multiplicative update approach can produce good solutions to (3), but that the rotation based algorithm finds a better solution to the LSD problem, in terms of the LSD objective, than the multiplicative update algorithm.

The LSD problem can also be approximated such that an alternating minimization approach may be used, as in Paatero and Tapper (1994); Paatero (1997). For example, we attempted to minimize $\|K - P^T Q\|_F + \lambda \|P - Q\|_F$ with the LSD constraints by alternately minimizing in P and Q, but the convergence was very poor. Other possibilities may be a greedy method using rank-one downdates (Biggs et al., 2008), or gradient descent (Paatero, 1999; Hoyer, 2004; Berry et al., 2007).

3.2 Rotation-based LSD Algorithm

We propose a rotation-based algorithm for solving (3) that we refer to as the *LSD algorithm*. The algorithm comprises three main steps: (i) initialize with an eigenvalue factorization (see Section 3.2.1), (ii) rotate the matrix factor to enforce the left-stochastic constraint, which puts each column of the matrix factor in the same plane as the probability simplex (see Section 3.2.2) and (iii) rotate again to enforce non-negativity constraints (see Section 3.2.3) which puts each column of the matrix factor inside the probability simplex. The algorithm is motivated by Theorem 3 which states that all matrix factors are related by a rotation. The complete algorithm is given in Algorithm 1 and Subroutines 1 and 2. Figure 2 summarizes the LSD algorithm for $k = 3$. The notation used in this section is summarized in Table 1.

Symbols	Description
$\|\cdot\|_2$	ℓ_2 norm
$\|\cdot\|_F$	Frobenius norm
$q_i \geq 0$	entry-wise inequality: $q_{ij} \geq 0$ for $j = 1, \ldots, k$
K'	scaled similarity matrix (see Section 3.2.1)
$\mathcal{H}$	a hyperplane (see Section 3.2.2)
$(Q)_\Delta$	matrix comprising columns of Q projected onto the probability simplex (see Section 3.2.2)
$\mathrm{SO}(k)$	group of $k \times k$ rotation matrices
ψ_u	embedding of $\mathrm{SO}(k-1)$ into $\mathrm{SO}(k)$ as the isotropy subgroup of $u \in \mathbb{R}^k$ (see Section 3.2.3)
π	a projection from $\mathbb{R}^{k\times k}$ to $\mathbb{R}^{(k-1)\times(k-1)}$; see (13)
$g^{(t)}$	rotation estimate after t^{th} iteration

Table 1: Notation used in Section 3.2

3.2.1 STEP 1: INITIALIZE WITH EIGENVALUE FACTORIZATION

A given similarity matrix is first scaled to get $K' = c^*K$ where c^* is given in Equation (10). Consider the eigendecomposition of the scaled similarity matrix, $K' = \sum_{i=1}^n \lambda_i v_i v_i^T$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$. The scaled similarity matrix is factorized as $K' \approx M^T M$, where $M = [\sqrt{\lambda_1} v_1 \; \sqrt{\lambda_2} v_2 \; \cdots \sqrt{\lambda_k} v_k]^T$ is a $k \times n$ real matrix comprising scaled eigenvectors corresponding to the top k eigenvalues. Note that this is where we require that the top k eigenvalues of the similarity matrix be positive. Furthermore, the matrix $M^T M$ is the best rank-k approximation of K' in terms of the Frobenius norm.

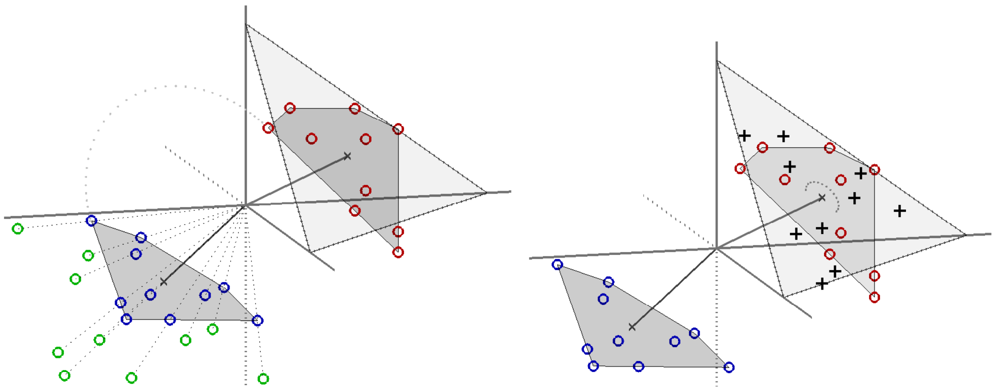
3.2.2 STEP 2: ENFORCE LEFT-STOCHASTIC CONSTRAINT

The n columns of M can be seen as points in k-dimensional Euclidean space, and the objective of the LSD algorithm is to rotate these points to lie inside the probability simplex in $\mathbb{R}^k$. However, the columns of M may not lie on a hyperplane in $\mathbb{R}^k$. Therefore, the next step is to project the columns of M onto the best fitting (least-squares) hyperplane, $\mathcal{H}$. Let m denote the normal to the hyperplane $\mathcal{H}$, and let $\tilde{M}$ denote the matrix obtained after projection of columns of M onto a hyperplane perpendicular to m and $1/\sqrt{k}$ units away from the origin (see Figure 2(a) for an illustration and Algorithm 1 for details).

Next the columns of $\tilde{M}$ are rotated by a rotation matrix R_s that rotates the unit vector $\frac{m}{\|m\|_2} \in \mathbb{R}^k$ about the origin to coincide with the unit vector $u = \frac{1}{\sqrt{k}}[1, \ldots, 1]^T$, which is normal to the probability simplex; that is, $R_s \frac{m}{\|m\|_2} = \frac{u}{\|u\|_2}$ (see Figure 2(a) for an illustration for the case where $k = 3$). The rotation matrix R_s is computed from a Givens rotation, as described in Subroutine 1. This rotation matrix acts on $\tilde{M}$ to give $Q = R_s \tilde{M}$, whose columns are points in $\mathbb{R}^k$ that lie on the hyperplane containing the probability simplex.

Special case ($k = 2$): For the special case of binary clustering with $k = 2$, the rotation matrix $R_s = U R_G U^T$, where R_G is the Givens rotation matrix

$$R_G = \begin{bmatrix} u^T m & -1 + (u^T m)^2 \\ 1 - (u^T m)^2 & u^T m \end{bmatrix},$$



(a) Rotate M to the simplex hyperplane

(b) Rotate about the unit vector until all points are inside the probability simplex

Figure 2: The proposed rotation-based LSD algorithm for a toy example with an LSDable K and $k=3$. Figure 2a shows the columns of M from Step 2, where $K' = M^T M$. The columns of M correspond to points in $\mathbb{R}^k$, shown here as green circles in the negative orthant. If K is LSDable, the green circles would lie on a hyperplane. We scale each column of M so that the least-squares fit of a hyperplane to columns of M is $1/\sqrt{k}$ units away from the origin (i.e., the distance of the probability simplex from the origin). We then project columns of M onto this hyperplane, mapping the green circles to the blue circles. The normal to this best-fit hyperplane is first rotated to the vector $u = \frac{1}{k}[1,\ldots,1]^T$ (which is normal to the probability simplex), mapping the blue circles to the red circles, which are the columns of Q in Step 3. Then, as shown in Figure 2b, we rotate the columns of Q about the normal u to best fit the points inside the probability simplex (some projection onto the simplex may be needed), mapping the red circles to black crosses. The black crosses are the columns of the solution P^*.

with $u = \frac{1}{\sqrt{2}}[1 \quad 1]^T$ and the unitary matrix $U = \begin{bmatrix} \frac{m}{\|m\|_2} & \frac{v}{\|v\|_2} \end{bmatrix}$ with $v = u - \frac{u^T m}{\|m\|_2^2} m$. We can then simply satisfy all LSD constraints for the matrix factor $Q = R_s \tilde{M}$ by projecting each column of Q onto the probability simplex (Michelot, 1986), that is, $\hat{P} = (Q)_\Delta$; we use the notation $(Q)_\Delta$ to denote the matrix obtained from a matrix Q by projecting each column of Q onto the probability simplex. For $k=2$, we do not need Step 3 which finds the rotation about the normal to the simplex that rotates each column of the matrix factor into the simplex. In this special case, there are only two possible rotations that leave the normal to the probability simplex in $\mathbb{R}^2$ invariant, corresponding to permuting the labels.

3.2.3 STEP 3: ENFORCE NON-NEGATIVITY CONSTRAINT

For $k > 2$, a final step is needed in which we rotate the columns of Q about u in the hyperplane containing the simplex to fit each column of Q into the simplex (see Figure 2(b) for an illustration). This requires estimating a rotation matrix, denoted R_u, that simultaneously rotates all points (i.e., columns of Q) into the probability simplex, thereby satisfying all LSD constraints. The matrix R_u

Algorithm 1: Rotation-based LSD Algorithm

```
Input: Similarity matrix $K \in \mathbb{R}^{n\times n}$, number of clusters $k$, number of iterations ITER
Compute $K' = c^*K$ where $c^*$ is given in Equation (10)
Compute rank-$k$ eigendecomposition $K'^{(k)} = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$
Compute $M = [\sqrt{\lambda_1}v_1 \ \sqrt{\lambda_2}v_2 \ \cdots \sqrt{\lambda_k}v_k]^T \ \in \mathbb{R}^{k\times n}$
Compute $m = (MM^T)^{-1}M\mathbf{1}_n$ (normal to least-squares hyperplane fit to columns of $M$)
Compute $\tilde{M} = \left(I - \frac{mm^T}{\|m\|_2}\right)M$ (project columns of $M$ onto the hyperplane normal to $m$ that passes through the origin)
Compute $\tilde{M} = \tilde{M} + \frac{1}{\sqrt{k}\|m\|_2}[m \ldots m]$ (shift columns $1/\sqrt{k}$ units in direction of $m$ )
Compute a rotation $R_s$=Rotate_Givens$(m,u)$ (see Subroutine 1 or if $k = 2$ the formula in Section 3.2.2)
Compute matrix $Q = R_s\tilde{M}$
If $k > 2$, compute $R_u$ = Rotate_Simplex$(K,Q,ITER)$ (see Subroutine 2),
else set $R_u = I$
Compute the (column-wise) Euclidean projection onto the simplex: $\hat{P} = (R_uQ)_\Delta$
Output: Cluster probability matrix $\hat{P}$
```

is learned using an incremental batch algorithm (adapted from Arora 2009b; Arora and Sethares 2010), as described in Subroutine 2. Note that it may not be possible to rotate each point into the simplex and therefore in such cases we would require a projection step onto the simplex after the algorithm for learning R_u has converged. Learning R_u is the most computationally challenging part of the algorithm, and we provide more detail on this step next.

Following Theorem 3(b), Subroutine 2 tries to find the rotation that best fits the columns of the matrix Q inside the probability simplex by solving the following problem

$$\begin{aligned} \underset{R\in \mathrm{SO}(k)}{\text{minimize}} \quad & \|K' - (RQ)_\Delta^T(RQ)_\Delta\|_F^2 \\ \text{subject to} \quad & Ru = u. \end{aligned} \tag{12}$$

The objective defined in (12) captures the LSD objective attained by the matrix factor $(RQ)_\Delta$. The optimization is over all $k \times k$ rotation matrices R that leave u invariant (the isotropy subgroup of u) ensuring that the columns of RQ stay on the hyperplane containing the probability simplex. This invariance constraint can be made implicit in the optimization problem by considering the following map from the set of $(k-1)\times(k-1)$ rotation matrices to the set of $k \times k$ rotation matrices (Arora, 2009a),

$$\begin{aligned} \psi_u : \mathrm{SO}(k-1) &\rightarrow \mathrm{SO}(k) \\ g &\mapsto R_{ue}^T \begin{bmatrix} g & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} R_{ue}, \end{aligned}$$

where $\mathbf{0}$ is a $k \times 1$ column vector of all zeros and R_{ue} is a rotation matrix that rotates u to $e = [0,\ldots,0,1]^T$, that is, $R_{ue}u = e$. The matrix R_{ue} can be computed using Subroutine 1 with inputs u and e. For notational convenience, we also consider the following map:

$$\begin{aligned} \pi : \quad \mathbb{R}^{k\times k} &\rightarrow \mathbb{R}^{(k-1)\times(k-1)} \\ \begin{bmatrix} A & b \\ c^T & d \end{bmatrix} &\mapsto A, \end{aligned} \tag{13}$$

where $b, c^T \in \mathbb{R}^k$ and d is a scalar.

It is easy to check that any rotation matrix R that leaves a vector $u \in \mathbb{R}^k$ invariant can be written as $\psi_u(g)$ for some rotation matrix $g \in \mathrm{SO}(k-1)$. We have thus exploited the invariance property of the isotropy subgroup to reduce our search space to the set of all $(k-1)\times(k-1)$ rotation matrices. The optimization problem in (12) is therefore equivalent to solving:

$$\underset{g\in\mathrm{SO}(k-1)}{\text{minimize}} \quad \|K' - (\psi_u(g)Q)_\Delta^T(\psi_u(g)Q)_\Delta\|_F^2 \quad . \tag{14}$$

We now discuss our iterative method for solving (14). Let $g^{(t)}$ denote the estimate of the optimal rotation matrix at iteration t.

Define matrices $X = \psi(g^{(t)})Q = [x_1 \ \ldots \ x_n]$ and $Y = (\psi(g^{(t)})Q)_\Delta = [y_1 \ \ldots \ y_n]$. Define $D \in \mathbb{R}^{n\times n}$ such that

$$D_{ij} = \begin{cases} 1, & q_i \geq 0, \ \mathbf{1}^T q_i = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Note that x_i represents the column q_i after rotation by the current estimate $\psi(g^{(t)})$ and y_i is the projection of x_i onto the probability simplex. We aim to seek the rotation that simultaneously rotates all q_i into the probability simplex. However, it may not be feasible to rotate all x_i into the probability simplex. Therefore, we update our estimate by solving the following problem:

$$g^{(t+1)} = \arg \min_{g\in\mathrm{SO}(k-1)} \sum_{i=1}^{n} D_{ii}\|y_i - \psi(g)x_i\|_2^2.$$

This is the classical orthogonal Procrustes problem and can be solved globally by singular value decomposition (SVD). Define matrix $T = \pi(YDX^T)$ and consider its SVD, $T = U\Sigma V^T$. Then the next iterate that solves (15) is given as $g^{(t+1)} = UV^T$.

Note that the sequence of rotation matrices generated by Subroutine 2 tries to minimize $\tilde{J}(R) = \|Q - (RQ)_\Delta\|_F$, rather than directly minimizing (12). This is a sensible heuristic because the two problems are equivalent when the similarity matrix is LSDable; that is, when the minimum value of $J(R)$ in (12) over all possible rotations is zero. Furthermore, in the non-LSDable case, the last step, projection onto the simplex, contributes towards the final objective $J(R)$, and minimizing $\tilde{J}(R)$ precisely reduces the total projection error that is accumulated over the columns of Q.

Subroutine 1: `Rotate_Givens` (Subroutine to Rotate a Unit Vector onto Another)

```
Input: vectors $m, u \in \mathbb{R}^k$
Normalize the input vectors, $m = \frac{m}{\|m\|_2}$, $u = \frac{u}{\|u\|_2}$.
Compute $v = u - \left(u^T m\right)\ m$. Normalize $v = \frac{v}{\|v\|_2}$.
Extend $\{m, v\}$ to a basis $U \in \mathbb{R}^{k \times k}$ for $\mathbb{R}^k$ using Gram-Schmidt orthogonalization.
Initialize $R_G$ to be a $k \times k$ identity matrix.
Form the Givens rotation matrix by setting:
        $(R_G)_{11} = (R_G)_{22} = u^T m,$
        $(R_G)_{21} = -(R_G)_{12} = u^T v.$
Compute $R_s = U R_G U^T$.
Output: $R_s$ (a rotation matrix such that $R_s \frac{m}{\|m\|_2} = \frac{u}{\|u\|_2}$).
```

Subroutine 2: `Rotate_Simplex` (Subroutine to Rotate Into the Simplex)

```
Input: Similarity matrix $K \in \mathbb{R}^{n \times n}$; matrix $Q = [q_1 \dots q_n] \in \mathbb{R}^{k \times n}$ with columns lying in the probability simplex hyperplane; maximum number of batch iterations $ITER$
Initialize $\psi(g_0), \hat{R}_u$ as $k \times k$ identity matrices.
Compute rotation matrix $R_{ue} = \texttt{Rotate\_Givens}(u, e) \in \mathbb{R}^k$ where $u = \frac{1}{\sqrt{k}}[1, \dots, 1]^T$ and $e = [0, \dots, 0, 1]^T \in \mathbb{R}^k$.
For $t = 1, 2, \dots, ITER$:
    Compute matrices $X = R_{ue}\psi(g_{t-1})Q$, $Y = R_{ue}(\psi(g_{t-1})Q)_\Delta$.
    Compute the diagonal matrix with $D_{ii} = 1 \iff x_i$ lies inside the simplex.
    If $\mathrm{trace}(D) = 0$, return $\hat{R}_u$.
    Compute $T = \pi(YDX^T)$ where $\pi$ is given in (13).
    Compute the SVD, $T = U\Sigma V^T$.
    Update $g^{(t)} = UV^T$.
    If $J(\psi(g^{(t)})) < J(\hat{R}_u)$, update $\hat{R}_u = \psi(g^{(t)})$.
Output: Rotation matrix $\hat{R}_u$.
```

3.3 Hierarchical LSD Clustering

For the special case of $k = 2$ clusters, the LSD algorithm described in Section 3.2 does not require any iterations. The simplicity and efficiency of the $k = 2$ case motivated us to explore a hierarchical binary-splitting variant of LSD clustering, as follows.

Start with all n samples as the root of the cluster tree. Split the n samples into two clusters using the LSD algorithm for $k = 2$, forming two leaves. Calculate the average within-cluster similarity of the two new leaves, where for the m-th leaf cluster C_m the within-cluster similarity is

$$W(C_m) = \frac{1}{n_m(n_m+1)} \sum_{\substack{i,j \in C_m, \\ i \le j}} K_{ij},$$

where C_m is the set of points belonging to the m-th leaf cluster, and n_m is the number of points in that cluster. Then, choose the leaf in the tree with the smallest average within-cluster similarity. Create a new similarity matrix composed of only the entries of K corresponding to samples in that leaf's cluster. Split the leaf's cluster into two clusters. Iterate until the desired k clusters are produced.

This top-down *hierarchical LSD clustering* requires running LSD algorithm for $k = 2$ a total of $k-1$ times. It produces k clusters, but does not produce an optimal $n \times k$ cluster probability matrix P. In Section 5, we show experimentally that for large k the runtime of hierarchical LSD clustering may be orders of magnitude faster than other clustering algorithms that produce similar within-cluster similarities.

3.4 An Alternating Minimization View of the LSD Algorithm

The rotation-based LSD algorithm proposed in Section 3.2 may be viewed as an "alternating minimization" algorithm. We can view the algorithm as aiming to solve the following optimization problem (in a slightly general setting),

$$\begin{array}{ll} \text{minimize}_{P,R} & \|XR - P\|_F^2 \\ \text{subject to} & R^T R = I \\ & P \in \mathcal{C}, \end{array} \tag{15}$$

where $P \in \mathbb{R}^{d \times k}$ and $R \in \mathbb{R}^{k \times k}$ are the optimization variables, R is an orthogonal matrix, $X \in \mathbb{R}^{d \times k}$ is the given data, and $\mathcal{C}$ is any convex set (for example in the LSD problem, it is the unit simplex). Geometrically, in this general problem the goal is to find an orthogonal transform that maps the rows of X into the set $\mathcal{C}$.

Unfortunately this problem is not jointly convex in P and R. A heuristic approach is to alternately fix one variable and minimize over the other variable, and iterate. This gives a general algorithm that includes LSD as a special case. The algorithm can be described as follows: At iteration k, fix P_k and solve the following problem for R,

$$\begin{array}{ll} \text{minimize}_{R} & \|XR - P_k\|_F^2 \\ \text{subject to} & R^T R = I, \end{array}$$

which is the well-known orthogonal Procrustes problem. The optimal solution is $R = UV^T$, where U, V are from the SVD of the matrix $X^T P$, that is, $X^T P = U \Sigma V^T$ (note that UV^T is also known as the "sign" matrix corresponding to $X^T P$). Then fix $R = R_k$ and solve for P,

$$\begin{array}{ll} \text{minimize}_{P} & \|XR_k - P\|_F^2 \\ \text{subject to} & P \in \mathcal{C}, \end{array}$$

where the optimal P is the Euclidean projection of XR_k onto the set $\mathcal{C}$. Update P as

$$P_{k+1} = \text{Proj}_{\mathcal{C}}(XR_k),$$

and repeat.

Computationally, the first step of the algorithm described above requires an SVD of a $k \times k$ matrix. The second step requires projecting d vectors of length k onto the set $\mathcal{C}$. In cases where this projection is easy to carry out, the above approach gives a simple and efficient heuristic for problem (15). Note that in the LSD algorithm, R is forced to be a rotation matrix (which is easy to do with a small variation of the first step). Also, at the end of each iteration k, the data X is also updated as $X_{k+1} = X_k R_k$, which means the rotation is applied to the data, and we look for further rotation to move our data points into $\mathcal{C}$. This unifying view shows how the LSD algorithm could be extended to other problems with a similar structure but with other constraint sets $\mathcal{C}$.

3.5 Related Clustering Algorithms

The proposed rotation-based LSD has some similarities to spectral clustering (von Luxburg, 2006; Ng et al., 2002) and to Perron cluster analysis (Weber and Kube, 2005; Deuflhard and Weber, 2005). Our algorithm begins with an eigenvalue decomposition, and then we work with the eigenvalue-scaled eigenvectors of the similarity matrix K. Spectral clustering instead acts on the eigenvectors of the graph Laplacian of K (normalized spectral clustering acts on the eigenvectors of the *normalized* graph Laplacian of K). Perron cluster analysis acts on the eigenvectors of row-normalized K; however, it is straightforward to show that these are the same eigenvectors as the normalized graph Laplacian eigenvectors and thus for the $k = 2$ cluster case, Perron cluster analysis and normalized spectral clustering are the same (Weber et al., 2004).

For $k > 2$ clusters, Perron cluster analysis *linearly maps* their $n \times k$ eigenvector matrix to the probability simplex to form a soft cluster assignment. In a somewhat similar step, we *rotate* a $n \times k$ matrix factorization to the probability simplex. Our algorithm is motivated by the model $K = P^T P$ and produces an exact solution if K is LSDable. In contrast, we were not able to interpret the Perron cluster analysis as solving a non-negative matrix factorization.

4. Experiments

We compared the LSD and hierarchical LSD clustering to nine other clustering algorithms: kernel convex NMF (Ding et al., 2010), unnormalized and normalized spectral clustering (Ng et al., 2002), k-means and kernel k-means, three common agglomerative linkage methods (Hastie et al., 2009), and the classic DIANA hierarchical clustering method (MacNaughton-Smith et al., 1964). In addition, we compared with hierarchical variants of the other clustering algorithms using the same splitting strategy as used in the proposed hierarchical LSD. We also explored how the proposed LSD algorithm compares against the multiplicative update approach adapted to minimize the LSD objective.

Details of how algorithm parameters were set for all experiments are given in Section 4.1. Clustering metrics are discussed in Section 4.2. The thirteen data sets used are described in Section 4.3.

4.1 Algorithm Details for the Experiments

For the LSD algorithms we used a convergence criterion of absolute change in LSD objective dropping below a threshold of 10^{-6}, that is, the LSD algorithms terminate if the absolute change in the LSD objective at two successive iterations is less than the threshold.

Kernel k-means implements k-means in the implicit feature space corresponding to the kernel. Recall that k-means clustering iteratively assigns each of the samples to the cluster whose mean is the closest. This only requires being able to calculate the distance between any sample i and the mean of some set of samples $\mathcal{J}$, and this can be computed directly from the kernel matrix as follows. Let ϕ_i be the (unavailable) implicit feature vector for sample i, and suppose we are not given ϕ_i, but do have access to $K_{ij} = \phi_i^T \phi_j$ for any i, j. Then k-means on the ϕ features can be implemented directly from the kernel using:

$$\begin{aligned} ||\phi_i - \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}}\phi_j||_2^2 &= \left(\phi_i - \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}}\phi_j\right)^T \left(\phi_i - \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}}\phi_j\right) \\ &= K_{ii} - \frac{2}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} K_{ji} + \frac{1}{|\mathcal{J}|^2}\sum_{j,\ell\in\mathcal{J}} K_{j\ell}. \end{aligned}$$

For each run of kernel k-means, we used 100 random starts and chose the result that performed the best with respect to the kernel k-means problem.

Similarly, when running the k-means algorithm as a subroutine of the spectral clustering variants, we used Matlab's `kmeans` function with 100 random starts and chose the solution that best optimized the k-means objective, that is, within-cluster scatter. For each random initialization, `kmeans` was run for a maximum of 200 iterations. For normalized spectral clustering, we used the Ng-Jordan-Weiss version (Ng et al., 2002).

For *kernel convex NMF* (Ding et al., 2010) we used the NMF Matlab Toolbox (Li and Ngom, 2011) with its default parameters. It initializes the NMF by running k-means on the matrix K, which treats the similarities as features (Chen et al., 2009a).

The top-down clustering method DIANA (DIvisive ANAlysis) (MacNaughton-Smith et al., 1964; Kaufman and Rousseeuw, 1990) was designed to take a dissimilarity matrix as input. We modified it to take a similarity matrix instead, as follows: At each iteration, we split the cluster with the smallest average within-cluster similarity. The process of splitting a cluster C into two occurs iteratively. First, we choose the point $x_1 \in C$ that has the smallest average similarity to all the other points in the cluster and place it in a new cluster C_{new} and set $C_{old} = C\backslash\{x_1\}$. Then, we choose the point in C_{old} that maximizes the difference in average similarity to the new cluster, as compared to the old; that is, the point x that maximizes

$$\frac{1}{|C_{new}|}\sum_{y\in C_{new}} K_{xy} - \frac{1}{|C_{old}|-1}\sum_{y\neq x, y\in C_{old}} K_{xy}. \tag{16}$$

We place this point in the new cluster and remove it from the old one. That is, we set $C_{new} = C_{new}\cup\{x\}$ and $C_{old} = C_{old}\backslash\{x\}$, where x is the point that maximizes (16). We continue this process until the expression in (16) is non-positive for all $y \in C_{old}$; that is, until there are no points in the old cluster that have a larger average similarity to points in the new cluster, as compared to remaining points in the old cluster.

Many of the similarity matrices used in our experiments are not positive semidefinite. Kernel k-means, LSD methods and kernel convex NMF theoretically require the input matrix to be a positive semidefinite (PSD) matrix, and so we clipped any negative eigenvalues, which produces the closest (in terms of the Frobenius norm) PSD matrix to the original similarity matrix[2] (see Chen et al. 2009a for more details on clipping eigenvalues in similarity-based learning).

4.2 Metrics

There is no single approach to judge whether a clustering is "good," as the goodness of the clustering depends on what one is looking for. We report results for four common metrics: within-cluster similarity, misclassification rate, perplexity, and runtime.

4.2.1 AVERAGE WITHIN-CLUSTER SIMILARITY

One common goal of clustering algorithms is to maximize the similarity between points within the same cluster, or equivalently, to minimize the similarity between points lying in different clusters. For example, the classic k-means algorithm seeks to minimize within-cluster scatter (or dissimilarity), unnormalized spectral clustering solves a relaxed version of the RatioCut problem, and the Shi-Malik version of spectral clustering solves a relaxed version of the NCut problem (von Luxburg, 2006). Here, we judge clusterings on how well they maximize the average of the within-cluster similarities:

$$\frac{1}{\sum_{m=1}^{k} n_m^2} \sum_{m=1}^{k} \left(\sum_{\substack{i,j \in C_m, \\ i \neq j}} K_{ij} + \sum_{i \in C_m} K_{ii} \right), \tag{17}$$

where n_m is the number of points in cluster C_m. This is equivalent to the min-cut problem.

4.2.2 MISCLASSIFICATION RATE

As in Ding et al. (2010), the misclassification rate for a clustering is defined to be the smallest misclassification rate over all permutations of cluster labels. Färber et al. (2010) recently argued that such external evaluations are "the best way for fair evaluations" for clustering, but cautioned, "Using classification data for the purpose of evaluating clustering results, however, encounters several problems since the class labels do not necessarily correspond to natural clusters." For example, consider the Amazon-47 data set (see Section 4.3 for details), where the given similarity between two samples (books) A and B is the (symmetrized) percentage of people who buy A after viewing B on Amazon. The given class labels are the 47 authors who wrote the 204 books. A clustering method asked to find 47 clusters might not pick up on the author-clustering, but might instead produce a clustering indicative of sub-genre. This is particularly dangerous for the divisive methods that make top-down binary decisions - early clustering decisions might reasonably separate fiction and non-fiction, or hardcover and paperback. Despite these issues, we agree with other researchers that misclassification rate considered over a number of data sets is a useful way to compare clustering algorithms. We use Kuhn's bipartite matching algorithm for computing the misclassification rate (Kuhn, 1955).

4.2.3 PERPLEXITY

An alternate metric for evaluating a clustering given known class labels is conditional perplexity. The conditional perplexity of the conditional distribution $P(L|C)$, of label L given cluster C, with

[2] Experimental results with the kernel convex NMF code were generally not as good with the full similarity matrix as with the nearest PSD matrix, as suggested by the theory.

conditional entropy $H(L|C)$, is defined to be $2^{H(L|C)}$. Conditional perplexity measures the average number of classes that fall in each cluster, thus the lower the perplexity the better the clustering. Arguably, conditional perplexity is a better metric than misclassification rate because it makes "soft" assignments of labels to the clusters.

4.2.4 RUNTIME

For runtime comparisons, all methods were run on machines with two Intel Xeon E5630 CPUs and 24G of memory. To make the comparisons as fair as possible, all algorithms were programmed in Matlab and used as much of the same code as possible. To compute eigenvectors for LSD and spectral clustering, we used the `eigs` function in Matlab, which computes only the k eigenvalues and eigenvectors needed for those methods.

4.2.5 OPTIMIZATION OF THE LSD OBJECTIVE

Because both our LSD algorithm and the multiplicative update algorithm seek to solve the LSD minimization problem (3), we compare them in terms of (3).

4.3 Data Sets

The proposed method acts on a similarity matrix, and thus most of the data sets used are specified as similarity matrices as described in the next subsection. However, to compare with standard k-means, we also considered two popular Euclidean data sets, described in the following subsection. Most of these data sets are publicly available from the cited sources or from `idl.ee.washington.edu/similaritylearning`.

4.3.1 NATIVELY SIMILARITY DATA SETS

We compared the clustering methods on eleven similarity data sets. Each data set provides a pairwise similarity matrix K and class labels for all samples, which were used as the ground truth to compute the misclassification rate and perplexity.

Amazon-47: The samples are 204 books, and the classes are the 47 corresponding authors. The similarity measures the symmetrized percent of people who buy one book after viewing another book on Amazon.com. For details see Chen et al. (2009a).

Aural Sonar: The samples are 100 sonar signals, and the classes are target or clutter. The similarity is the average of two humans' judgement of the similarity between two sonar signals, on a scale of 1 to 5. For details see Philips et al. (2006).

Face Rec: The samples are 945 faces, and the classes are the 139 corresponding people. The similarity is a cosine similarity between the integral invariant signatures of the surface curves of the 945 sample faces. For details see Feng et al. (2007).

Internet Ads: The samples are 2359 webpages (we used only the subset of webpages that were not missing features), and the classes are advertising or not-advertising. The similarity is the Tversky similarity of 1556 binary features describing a webpage, which is negative for many pairs of webpages. For details see the UCI Machine Learning Repository and Cazzanti et al. (2009).

MIREX: The samples are 3090 pieces of music, and the classes are ten different musical genres. The similarity is the average of three humans' fine-grained judgement of the audio similarity of a pair of samples. For details see the Music Information Retrieval Evaluation eXchange (MIREX) 2007.

MSIFT: The samples are 477 images, class labels are nine scene types, as labeled by humans. The similarity is calculated from the multi-spectral scale-invariant feature transform (MSIFT) descriptors (Brown and Susstrunk, 2011) of two images by taking the average distance $\overline{d}$ between all pairs of descriptors for the two images, and setting the similarity to $e^{-\overline{d}}$. For details see Brown and Susstrunk (2011).

Patrol: The samples are 241 people, and the class labels are the eight units they belong to. The binary similarity measures the symmetrized event that one person identifies another person as being in their patrol unit. For details see Driskell and McDonald (2008).

Protein: The samples are 213 proteins, and four biochemically relevant classes. The similarity is a sequence-alignment score. We used the pre-processed version detailed in Chen et al. (2009a).

Rhetoric: The samples are 1924 documents, and the class labels are the eight terrorist groups that published the documents. The similarity measures KL divergence between normalized histograms of 173 keywords for each pair of documents. Data set courtesy of Michael Gabbay.

Voting: The samples are 435 politicians from the United States House of Representatives, and the class label is their political party. The similarity measures the Hamming similarity of sixteen votes in 1984 between any two politicians. For details see the UCI Machine Learning Repository.

Yeast: The samples are 2222 genes that have only one of 13 biochemically relevant class labels. The similarity is the Smith-Waterman similarity between different genes. For details see Lanckriet et al. (2004).

4.3.2 NATIVELY EUCLIDEAN DATA SETS

In order to also compare similarity-based clustering algorithms to the standard k-means clustering algorithm (Hastie et al., 2009), we used the standard MNIST and USPS benchmark data sets, which each natively consist of ten clusters corresponding to the ten handwritten digits 0-9. We subsampled the data sets to 600 samples from each of the ten classes, for a total of 6000 samples. We compute the locally translation-invariant features proposed by Bruna and Mallat (2011) for each digit image. The k-means algorithm computes the cluster means and cluster assignments using the features directly, whereas the similarity-based clustering algorithms use the RBF (radial basis function) kernel to infer the similarities between a pair of data points. The bandwidth of the RBF kernel was tuned for the average within cluster similarity on a small held-out set. We tuned the kernel bandwidth for the kernel k-means algorithm and the used the same bandwidth for all similarity-based algorithms. Note that different bandwidths yield different similarity matrices and the resulting average within cluster similarities (computed using Equation (17)) are not directly comparable for two different values of bandwidths. Therefore, we picked the kernel bandwidth that maximized the average within-cluster-similarity in the original feature space (Hastie et al., 2009).

5. Results

Results were averaged over 100 different runs,[3] and are reported in Table 2 (LSD objective minimization), Table 3 (average within-cluster similarity), Table 4 (perplexity), Table 5 (misclassifica-

[3] For some of the clustering algorithms, such as the linkages, there is no algorithmic randomness, but ties in the linkage values were broken arbitrarily.

tion rate). The results on the MNIST and the USPS data sets are reported in Table 6. Runtimes are reported in Table 7 (runtimes), and while averaged over 100 different runs, the runtime results were highly variable, and so we consider them comparable only in terms of order of magnitude.

5.1 Comparison of Rotation-based and Multiplicative Update Approaches to Minimize the LSD Objective

We compared the proposed rotation-based algorithm to the multiplicative-update approach for solving for the LSD optimization problem given in (3), as shown in Table 2. The resulting objective function values for the two algorithms were generally similar, with the LSD algorithm finding a better solution for ten of the eleven data sets.

In terms of within-cluster similarity, the multiplicative update approach was slightly better or tied with the rotation-based algorithm on all eleven similarity data sets. However, in terms of misclassification rate and perplexity, the rotation-based algorithm was almost always better, implying that the rotation-based method is picking up legitimate structure in the data sets even if it is not doing as well at maximizing the within-cluster similarity.

	LSD (Rotation-based algorithm)	LSD (Multiplicative-update algorithm)
Amazon47	**818**	882
AuralSonar	**17**	18
FaceRec	**18**	475
InternetAds	60093	**58093**
Mirex07	**632**	766
MSIFTavg	**14**	**14**
Patrol	**19**	22
Protein	**41**	43
Rhetoric	**23554**	25355
Voting	**27**	28
YeastSW13Class	**559**	**559**

Table 2: Comparison of optimized LSD objective values.

5.2 Comparison of LSD to Other Clustering Algorithms

The LSD model for clustering performed well. Of the 15 clustering methods compared on within-cluster similarity, LSD using the multiplicative-update minimization was the best (or tied for best) on nine of the eleven data sets. LSD using the rotation-based minimization had the best misclassification rate and perplexity most often of all the methods over the total of thirteen data sets.

The runtimes for the LSD algorithms were similar to the kernel convex NMF and spectral clustering methods.

5.3 Results for Hierarchical LSD

The most surprising result of our experiments was the good performance of the hierarchical methods in terms of all metrics considered. We implemented the hierarchical LSD to take advantage of

	Amaz47	AuralSon	FaceRec	Int Ads	Mirex07	MSIFT	Patrol	Protein	Rhetoric	Voting	Yeast
# classes	47	2	139	2	10	9	8	4	8	2	13
# points	204	100	945	2359	3090	477	241	213	1924	435	2222
LSD (M-Upd.)	**18.83**	**0.44**	0.88	-11.31	**0.05**	**0.38**	**0.09**	**0.51**	**36.13**	**0.79**	**0.66**
LSD (Rotation)	14.56	**0.44**	0.86	**-9.60**	0.03	**0.38**	0.08	0.49	**36.13**	**0.79**	**0.66**
Hier. LSD	16.20	**0.44**	**1.00**	-12.25	0.04	**0.38**	0.07	0.46	33.93	**0.79**	0.58
Kernel Conv. NMF	13.74	**0.44**	**1.00**	-10.19	0.03	**0.38**	0.08	0.49	35.08	**0.79**	0.64
Hier. Kernel Conv. NMF	15.48	**0.44**	**1.00**	-10.16	0.04	**0.38**	0.08	**0.51**	35.99	**0.79**	0.64
Kernel k-Means	1.15	0.43	**1.00**	-9.97	0.01	0.37	0.02	0.41	35.03	**0.79**	0.52
Hier. Kernel k-Means	3.04	0.43	**1.00**	-9.97	0.01	0.37	0.03	0.38	34.76	**0.79**	0.53
Unnorm. Spec.	14.35	0.34	0.92	-11.38	0.01	0.37	0.08	0.30	34.89	**0.79**	0.52
Hier. Unnorm. Spec.	14.97	0.34	**1.00**	-11.38	0.01	0.37	0.08	0.30	34.89	**0.79**	0.52
Norm. Spec	15.38	**0.44**	0.89	**-9.60**	0.01	**0.38**	0.08	**0.51**	35.00	**0.79**	**0.66**
Hier. Norm. Spec.	15.79	**0.44**	**1.00**	-9.70	0.02	**0.38**	0.07	**0.51**	35.67	**0.79**	0.62
Sing. Link.	10.71	0.33	**1.00**	-11.32	0.01	0.37	0.08	0.29	34.87	0.55	0.52
Comp. Link.	1.41	0.33	**1.00**	-11.12	0.01	0.37	0.01	0.35	34.84	0.77	0.56
Avg. Link.	14.74	0.36	**1.00**	-11.23	0.01	0.37	0.08	0.31	34.89	0.77	0.56
DIANA	7.89	**0.44**	0.98	-10.61	0.01	0.37	0.08	0.35	34.91	**0.79**	0.62

Table 3: Average within-cluster similarity.

the simplicity and efficiency of the $k = 2$ case of the rotation-based LSD algorithm. Because the other clustering algorithms do not have fast special cases for $k = 2$, hierarchical variants of these methods do not offer increased efficiency, but we compared to them for completeness. Surprisingly, the hierarchical variants generally did not degrade performance. For example, the hierarchical normalized spectral clustering is as good or tied with normalized spectral clustering on eight of the eleven similarity data sets.

The hierarchical LSD method performed consistently fast, and achieved good results in terms of within-cluster similarity and misclassification rate. For example, for the Face Recognition data set (with $k = 139$) the runtime of hierarchical LSD is an order of magnitude faster than the LSD rotation-based algorithm, and achieves the highest average within-cluster similarity.

6. Conclusions, Further Discussion, and Open Questions

A number of NNMF models and algorithms have been proposed for clustering. In this paper, we proposed a left-stochastic NNMF model for clustering, based on relaxing an ideal kernel model. We showed that the proposed LSD NNMF can be effectively approximately solved using a standard multiplicative update approach, but that the same or better objective values can be reached using a novel rotation-based algorithm. For $k = 2$, the proposed LSD algorithm provides a unique solution without iterations or the need for multiple starts. This fact motivated a fast hierarchical LSD clustering for problems where the number of clusters desired is large. For most data sets, the proposed LSD clustering and hierarchical LSD were top performers.

We showed that the set of possible LSD clusterings is related by rotations and gave conditions for when the LSD clustering is unique. This property makes it trivial to discover *multiple clusterings* (Niu et al., 2010) by simply rotating an LSD solution. In this paper, we only considered converting the LSD solution P^* to a clustering by classifying each sample to the highest-probability cluster. However, an interesting advantage of LSD is that samples could instead be assigned in rank order

	Amaz47	AuralSon	FaceRec	Int Ads	Mirex07	MSIFT	Patrol	Protein	Rhetoric	Voting	Yeast
# classes	47	2	139	2	10	9	8	4	8	2	13
# points	204	100	945	2359	3090	477	241	213	1924	435	2222
LSD (M-Upd.)	1.46	1.46	23.00	1.56	5.99	7.13	1.33	1.46	7.24	1.33	6.76
LSD (Rotation)	**1.36**	**1.44**	24.83	1.53	**4.86**	7.11	**1.23**	1.70	**6.24**	1.33	**6.51**
Hier. LSD	1.48	**1.44**	1.25	1.54	5.11	6.63	1.84	1.67	**6.24**	1.33	7.16
Kernel Conv. NMF	1.41	1.49	1.65	1.55	4.89	5.33	1.36	1.65	7.12	1.34	**6.51**
Hier. Kernel Conv. NMF	1.44	1.49	1.24	1.54	5.14	**5.28**	1.32	1.42	7.31	1.34	6.78
Kernel k-Means	7.78	1.56	1.49	1.53	8.56	7.73	3.41	2.18	6.96	1.33	8.46
Hier. Kernel k-Means	3.09	1.55	1.22	1.53	8.63	8.67	2.82	2.50	7.46	1.33	8.19
Unnorm. Spec.	1.37	1.99	17.61	1.56	9.81	8.67	1.24	3.60	7.43	1.33	8.76
Hier. Unnorm. Spec.	1.44	1.99	1.26	1.56	9.79	8.67	1.24	3.60	7.43	1.33	8.49
Norm. Spec	1.37	**1.44**	1.34	1.53	8.41	5.29	1.24	1.37	7.15	1.33	6.61
Hier. Norm. Spec.	1.45	**1.44**	1.20	1.54	7.38	5.55	1.43	**1.35**	7.24	1.33	6.95
Sing. Link.	1.61	1.99	1.06	1.56	9.87	8.67	1.24	3.68	7.43	1.94	8.78
Comp. Link.	5.19	1.96	**1.04**	1.54	9.87	8.51	7.06	2.72	7.39	**1.18**	7.51
Avg. Link.	1.37	1.92	**1.04**	1.55	9.87	8.67	**1.23**	3.39	7.43	1.22	7.30
DIANA	2.18	1.52	11.60	**1.47**	9.87	8.67	**1.23**	2.58	7.46	1.33	6.71

Table 4: Perplexity.

	Amaz47	AuralSon	FaceRec	Int Ads	Mirex07	MSIFT	Patrol	Protein	Rhetoric	Voting	Yeast
# classes	47	2	139	2	10	9	8	4	8	2	13
# points	204	100	945	2359	3090	477	241	213	1924	435	2222
LSD (M-Upd.)	0.38	0.15	0.89	0.44	**0.66**	0.77	0.16	0.13	0.80	0.10	0.77
LSD (Rotation)	**0.24**	**0.14**	0.87	0.35	**0.66**	0.77	0.09	0.33	**0.75**	0.10	0.76
Hier. LSD	0.36	**0.14**	0.19	0.46	**0.66**	0.74	0.29	0.37	**0.75**	0.10	0.76
Kernel Conv. NMF	0.26	**0.14**	0.28	0.26	**0.66**	0.69	0.17	0.25	0.81	0.10	0.76
Hier. Kernel Conv. NMF	0.31	**0.14**	0.19	0.26	**0.66**	0.67	0.14	0.13	0.80	0.10	0.74
Kernel k-Means	0.69	0.20	0.31	0.29	0.83	0.80	0.55	0.47	0.77	0.10	0.78
Hier. Kernel k-Means	0.50	0.19	0.16	0.29	0.83	0.88	0.44	0.48	0.78	0.10	0.78
Unnorm. Spec.	0.25	0.49	0.92	0.17	0.88	0.88	0.09	0.65	0.78	0.09	0.78
Hier. Unnorm. Spec.	0.29	0.49	0.17	0.17	0.88	0.88	0.09	0.65	0.78	0.09	0.77
Norm. Spec	0.26	**0.14**	0.23	0.31	0.81	**0.62**	0.09	0.11	0.81	0.10	0.78
Hier. Norm. Spec.	0.31	**0.14**	0.19	0.31	0.78	0.66	0.18	**0.10**	0.80	0.10	0.75
Sing. Link.	0.30	0.49	0.04	0.16	0.88	0.88	0.06	0.66	0.78	0.38	0.78
Comp. Link.	0.69	0.47	**0.03**	0.16	0.88	0.87	0.78	0.59	0.79	**0.04**	0.82
Avg. Link.	0.25	0.41	**0.03**	0.16	0.88	0.88	**0.05**	0.62	0.78	0.05	0.72
DIANA	0.36	0.16	0.87	**0.13**	0.88	0.88	0.11	0.55	0.78	0.10	**0.71**

Table 5: Misclassification rates.

to different clusters to produce a clustering with a desired number of samples in each cluster. This approach can also be used to produce a set of multiple clusterings.

LSD produces cluster probabilities P^*, and this feature was not explored in this paper. Experimentally, tests are needed to assess and compare the performance of the actual cluster probabilities (rather than threshold them to form a hard clustering). LSD's probabilistic model does not explicitly assume a particular distribution, but there may be a relationship between the goodness of the LSD model and specific distributions. A related open question is why LSD is better suited to some data sets than others. For example, we know that k-means tends to perform well on well-separated

	USPS		MNIST	
	Perplexity	Misclassification Rate	Perplexity	Misclassification Rate
k-Means	2.23	0.34	2.98	0.47
LSD (M-Upd.)	2.03	0.30	2.83	0.42
LSD (Rotation)	**1.97**	**0.27**	**2.27**	**0.34**
Hier. LSD	2.38	0.41	2.89	0.45
Kernel Conv. NMF	2.23	0.38	2.82	0.42
Hier. Kernel Conv. NMF	2.22	0.34	2.83	0.42
Kernel k-Means	2.20	0.33	2.90	0.46
Hier. Kernel k-Means	2.20	0.33	3.01	0.48
Unnorm. Spec.	9.07	0.81	8.75	0.81
Hier. Unnorm. Spec.	9.05	0.80	8.75	0.80
Norm. Spec	2.24	0.36	3.30	0.52
Hier. Norm. Spec.	2.23	0.32	3.18	0.50
Sing. Link.	9.20	0.83	8.80	0.81
Comp. Link.	3.38	0.53	5.06	0.66
Avg. Link.	3.89	0.61	7.24	0.73
DIANA	4.02	0.65	3.64	0.55

Table 6: Clustering results for two natively Euclidean benchmark data sets. K-means acts on the Euclidean features, all the other algorithms use the same RBF similarity (see text for details).

	Amaz47	AuralSon	FaceRec	Int Ads	Mirex07	MSIFT	Patrol	Protein	Rhetoric	Voting	Yeast
# classes	47	2	139	2	10	9	8	4	8	2	13
# points	204	100	945	2359	3090	477	241	213	1924	435	2222
LSD (M-Upd.)	10	0	335	512	761	60	0	1	724	1	756
LSD (Rotation)	11	0	1672	0	40	2	3	12	60	0	883
Hier. LSD	1	0	2	1	4	0	0	0	1	0	1
Kernel Conv. NMF	14	1	552	1292	767	3	3	6	589	47	235
Hier. Kernel Conv. NMF	4	1	389	1297	385	50	1	10	1396	46	1705
Kernel k-Means	1154	9	5471	26452	84831	174	114	49	131131	44	330502
Hier. Kernel k-Means	267	9	783	262110	305100	361	109	69	314500	44	1318310
Unnorm. Spec.	5	0	529	111	13	14	1	1	52	0	258
Hier. Unnorm. Spec.	6	0	19	111	21	3	1	1	29	0	46
Norm. Spec	5	0	51	209	103	5	1	0	13	0	32
Hier. Norm. Spec.	5	0	16	209	19	2	1	0	5	0	6
Sing. Link.	8	1	796	12319	27544	103	13	9	6625	78	10363
Comp. Link.	9	1	849	13179	29279	109	14	10	7242	83	11155
Avg. Link.	21	3	2143	33892	75624	279	36	25	18367	212	27834
DIANA	1	0	58	21	5	0	2	0	8	5	301

Table 7: Runtimes in seconds.

and convex clusters, and poorly on data sets where the convex hulls of different clusters intersect. Currently, we lack a similar intuition for LSD, and this question may be intimately related to the question of what the LSD model implies in terms of the geometric distribution of samples among clusters.

Our model assumed a constant scale factor c such that $K = P^T P/c$ for a given similarity matrix K. A different scaling could be better. For example, one could normalize K by the inverse degree matrix D^{-1}, as done in Perron cluster analysis (Weber et al., 2004), but $D^{-\frac{1}{2}}KD^{-\frac{1}{2}}$ will always be right-stochastic, and $P^T P$ is not generally right-stochastic, so the resulting model $D^{-\frac{1}{2}}KD^{-\frac{1}{2}} = P^T P$ is not sensible. However, some other non-constant scaling could lead to a more sensible model.

We reported results in terms of standard clustering metrics. However, clustering is often a useful step in a bigger processing chain, and not an end to itself. For example, Nelson and Gupta (2007) clustered receivers to initialize algorithms that estimate the locations of multiple transmitters. Recently, Hanusa et al. (2011) clustered likelihood surfaces in order to fuse returns from multiple receivers in tracking. In such cases, the right metric for the clustering methods is the end-to-end application-specific metric; in these examples, the transmitter estimation and the multi-static tracking performance.

The rotation-based matrix factorization approach proposed in this paper can in fact be applied in a much more general setting. For example, it can be used as a heuristic for the completely positive matrix factorization problem, if we enforce only nonnegativity, and drop the $P^T\mathbf{1} = \mathbf{1}$ constraint. More broadly, the approach can be extended to any problem where the goal is to obtain vectors that lie in some *convex cone*, given the matrix of their pairwise inner products. That is, given the *Gram matrix* $K \in \mathbb{R}^{n\times n}$ where $K_{ij} = x_i^T x_j$, we want to find the underlying vectors $x_i \in \mathbb{R}^k$, $i = 1,\dots,n$, knowing that all x_i belong to the convex cone $\mathcal{C}$ (see problem (15), in Section 3.4, which describes the general algorithm idea). Similar to the LSD algorithm, this general iterative algorithm will start by factorizing the Gram matrix to obtain an initial set of vectors (we assume the correct dimension of the x_i are known) and seek a rotation matrix that maps these vectors into the cone $\mathcal{C}$. At every iteration, the algorithm will project the vectors onto the cone, then update the rotation matrix accordingly as discussed in Section 3.4. Thus the LSD algorithm approach could be applied to other problems with a similar structure but with other constraint sets $\mathcal{C}$. For example, one might want to find an orthogonal transformation that maps a set of matrices into the positive semidefinite cone (this arises, for example, when dealing with estimates of a set of covariance matrices). This is a topic for future exploration.

Another future direction is to explore scalable algorithms for similarity-based clustering. All of the similarity-based algorithms studied in this paper are computationally expensive. For instance, the proposed rotation-based LSD algorithm has space complexity of $O(n^2)$ and computational complexity of $O(n^2k)$. For large data sets the similarity-based algorithms studied in this paper become computationally infeasible. One approach to address the scalability issue is to consider stochastic approximation algorithms that process small number of entries in the similarity matrix at each iteration. Note that the first step in the rotation-based LSD algorithm involves finding a rank-k SVD of a given similarity matrix. For large similarity matrices, we can employ an incremental SVD algorithm that processes a single column of the matrix at each iteration (Arora et al., 2012). Such stochastic approximation approaches for large-scale matrix factorization problems in the kernel setting have recently been shown to be useful for machine learning tasks (Arora and Livescu, 2012, 2013).

Acknowledgments

We thank James Chen, Sergey Feldman, Bela Frigyik and Kristi Tsukida for helpful discussions. This research was supported by a United States PECASE Award, the United States Office of Naval Research, NSF Award #1027812 and and the NSF CAREER Award ECCS-0847077.

Appendix A. Proof of Proposition 1

Proof Note that the lower bound of (4) is zero; we will show that the LSD clustering solution P_* achieves this lower bound. Let $J(F) = \|\Phi(X) - FP\|_F^2$, then a stationary point with respect to F occurs at $\partial J(F)/\partial F = 0$, which is at $F_* = \Phi(X)P^T(PP^T)^{-1}$. Note that $(PP^T)^{-1}$ exists because P is full rank. With this F_*, (4) can be written:

$$\begin{aligned}
&\arg\min_{P:P\geq 0, P^T1=1} \|\Phi(X) - \phi(X)P^T(PP^T)^{-1}P\|_F^2 \\
\equiv\ &\arg\min_{P:P\geq 0, P^T1=1} \mathrm{tr}(K) - 2\mathrm{tr}(KP^T(PP^T)^{-1}P) + \mathrm{tr}(KP^T(PP^T)^{-1}P) \\
\equiv\ &\arg\min_{P:P\geq 0, P^T1=1} \mathrm{tr}(K) - \mathrm{tr}(KP^T(PP^T)^{-1}P) \qquad (18)
\end{aligned}$$

Consider any P_* that solves the LSD problem such that $K = P_*^T P_*$, then the objective function in (18) becomes:

$$\begin{aligned}
J(P^*) &= \mathrm{tr}(K) - \mathrm{tr}(P_*^T P_* P_*^T (P_* P_*^T)^{-1} P_*) \\
&= \mathrm{tr}(K) - \mathrm{tr}(P_*^T P_*) \\
&= \mathrm{tr}(K) - \mathrm{tr}(K) \\
&= 0.
\end{aligned}$$

Since the LSD solution P_* achieves the lower bound of (18), it must be a minimizer. ■

Appendix B. Proof of Proposition 2

Proof By definition of LSDable, there exists a scalar $c \in \mathbb{R}_+$ and a left-stochastic matrix $P \in \mathbb{R}^{k\times n}$ such that $cZ^TZ = P^TP$. Then, Theorem 3(a) states that there exists an orthogonal matrix $R \in \mathbb{R}^{k\times k}$ such that

$$R(\sqrt{c}Z) = P. \qquad (19)$$

And by the left-stochasticity of P, we know that

$$P^T\mathbf{1}_k = \mathbf{1}_n. \qquad (20)$$

Substituting (19) into (20):

$$\begin{aligned}
&(R(\sqrt{c}Z))^T\mathbf{1}_k = \mathbf{1}_n, \\
\Rightarrow\quad &\sqrt{c}Z^TR^T\mathbf{1}_k = \mathbf{1}_n, \\
\Rightarrow\quad &\sqrt{c}ZZ^TR^T\mathbf{1}_k = Z\mathbf{1}_n, \\
\Rightarrow\quad &R^T\mathbf{1}_k = \frac{1}{\sqrt{c}}(ZZ^T)^{-1}Z\mathbf{1}_n, \qquad (21)
\end{aligned}$$

where $(ZZ^T)^{-1}$ exists because Z is full-rank.

Take the ℓ_2 norm of both sides of (21) to conclude:

$$\begin{aligned} \|R^T\mathbf{1}_k\|_2 &= \frac{\|(ZZ^T)^{-1}Z\mathbf{1}_n\|_2}{\sqrt{c}}, \\ \Rightarrow \qquad c &= \frac{\|(ZZ^T)^{-1}Z\mathbf{1}_n\|_2^2}{\|R^T\mathbf{1}_k\|_2^2}, \\ \Rightarrow \qquad c &= \frac{\|(ZZ^T)^{-1}Z\mathbf{1}_n\|_2^2}{k}, \end{aligned} \tag{22}$$

because the ℓ_2 norm is unitary invariant and $\|\mathbf{1}_k\|_2 = \sqrt{k}$.

Next we show that c^* does not depend on any particular factorization of the LSDable matrix Z^TZ. Let $Q \in \mathbb{R}^{k\times n}$ be any matrix such that $Q^TQ = Z^TZ$. Then Theorem 3(a) implies that there exists a rotation matrix $\tilde{R}$ such that $\tilde{R}Q = Z$. Substituting $Z = \tilde{R}Q$ in (22),

$$\begin{aligned} c &= \frac{\|(\tilde{R}QQ^T\tilde{R}^T)^{-1}\tilde{R}Q\mathbf{1}_n\|_2^2}{k}, \\ &= \frac{\|\tilde{R}^{-T}(\tilde{R}QQ^T)^{-1}\tilde{R}Q\mathbf{1}_n\|_2^2}{k}, \\ &= \frac{\|\tilde{R}(QQ^T)^{-1}\tilde{R}^{-1}\tilde{R}Q\mathbf{1}_n\|_2^2}{k}, \\ &= \frac{\|(QQ^T)^{-1}Q\mathbf{1}_n\|_2^2}{k}, \end{aligned}$$

where we used the fact that $(AB)^{-1} = B^{-1}A^{-1}$ if matrices A and B are invertible in the second and third steps, and unitary invariance of the ℓ_2 norm in the last equality. ■

Appendix C. Proof of Theorem 3

Proof (a) If K has two decompositions $K = P^TP = Q^TQ$, for $P, Q \in \mathbb{R}^{m\times n}$ then $K_{ij} = P_i^TP_j = Q_i^TQ_j$, that is, the inner product between the ith and jth columns of P and Q are the same. The linear transformation that preserves inner-products for all $P_i \in \mathbb{R}^m$ is an orthogonal transformation $R \in O(m)$. The transformation is unique because if there were two different elements $R_1, R_2 \in O(m)$, $R_1 \neq R_2$, such that $P = R_1Q$ and $P = R_2Q$, then $Q = R_1^TR_2Q$, which would imply $R_1^TR_2 = I$. Multiplying both sides from the left by R_1, we get $R_2 = R_1$, which is a contradiction.

(b) Let $\mathcal{H} = \{(x_1,\ldots,x_k) \in \mathbb{R}^k | \sum_{j=1}^k x_k = 1\}$ denote the hyperplane that contains the simplex Δ^k. Let u denote the normal to the simplex Δ^k. Then the proof follows from the fact that the subset of orthogonal transformations (from part(a)) that map the hyperplane $\mathcal{H}$ onto itself are the transformations that leave u invariant, that is, the stabilizer subgroup of u. ■

Appendix D. Proof of Theorem 4

Proof Let SO(k) denote the special orthogonal group in $\mathbb{R}^k$, that is, the set of all rotation matrices of size $k\times k$ and let μ denote the Haar measure associated with SO$(k-1)$. Given a vector $x \in \mathbb{R}^k$, the stabilizer subgroup of x is defined to be the set of all rotations that leave x invariant. From Theorem 1(b), we know that all left-stochastic decompositions are related by orthogonal transformations that leave $u = [1, \ldots, 1]^T \in \mathbb{R}^k$ fixed. Since we are interested in clusterings that are unique up to a permutation of labels, and the orthogonal group modulo reflections is isomorphic to the rotation group, all left-stochastic decompositions are characterized by the stabilizer subgroup of u, given by

$$H_u = \{R \in \mathrm{SO}(k) | Ru = u\}.$$

Let P be any LSD factor of a given similarity matrix K, and let $G(P) \subseteq H_u$ be the set of all rotations that gives feasible LSD factors for K, that is,

$$G(P) = \{R \in H_u | (RP)_j \in \Delta^k \text{ for } j = 1, \ldots, n\},$$

where Δ^k is the probability simplex in $\mathbb{R}^k$. Note that the set $G(P)$ is not empty as it always contains the identity matrix. We consider two partitions of $G(P)$: the first partition $G(P) = \cup_{i \in I^{(LSD)}} G_i^{(LSD)}(P)$ induced by the equivalence relationship that $R', R'' \in G_i^{(LSD)}(P)$ if and only if both $R'P$ and $R''P$ give the same LSD clustering; the second partition $G(P) = \cup_{i \in I^{(conn)}} G_i^{(conn)}(P)$ induced by the equivalence relationship that $R', R'' \in G_i^{(conn)}(P)$ if and only if there is a continuous connected path $\{R_t | t \in [0,1]\}$ such that $R_0 = R', R_1 = R''$ and $R_t \in G_i^{(conn)}$ for all $t \in [0,1]$. Note that neither partition is empty because $G(P)$ is not empty, and therefore the following volumes are well defined:

$$\begin{aligned} \alpha &= \sup_{i \in I^{(conn)}} \mu(G_i^{(conn)}(P)), \\ \beta &= \inf_{i \in I^{(LSD)}} \mu(G_i^{(LSD)}(P)). \end{aligned}$$

Next, note that $G(P) = H_u$ if and only if $\|P_j\|_2 \leq \frac{1}{\sqrt{k-1}}$ for all $j = 1, \ldots, n$ (i.e., columns of P, seen as points in $\mathbb{R}^k$, lie inside the largest sphere inscribed inside the simplex), in which case $I^{(conn)} = \{1\}$ with $G_1^{(LSD)}(P) = H_u$, giving $\alpha = \mu(H_u) = 1$. Clearly, we cannot have a unique clustering in this case as we can arbitrarily rotate the columns of P about the centroid of the simplex without violating any LSD constraints and resulting in arbitrary LSD clusterings. Furthermore, an upper bound on β is $1/k$ because of the symmetry about the centroid of the simplex and so $\alpha \not< \beta$; the uniqueness condition therefore takes care of this trivial scenario.

If the condition $\|P_j\|_2 \leq \frac{1}{\sqrt{k-1}}$ does not hold for all columns of P, then there is a non-trivial partition of $G(P)$ into connected components and therefore $\alpha < 1$. The result then follows by a simple observation that the clustering changes while still being a valid LSD if and only if we can rotate a point into a different clustering without rotating any points out of the simplex, that is, there exist sets $G_i^{(LSD)}(P)$ and $G_j^{(conn)}(P)$ in the two partitions such that $G_i^{(LSD)}(P) \subset G_j^{(conn)}(P)$ but then $\alpha \geq \beta$. Therefore, a sufficient condition for the LSD clustering to be unique (up to re-labeling) is if $\alpha < \beta$. ■

Appendix E. Proof of Theorem 5

Proof The error bound in (11) involves three terms: the first term captures the perturbation of the eigenvalues and scales linearly with ε; the second term involves $\|K^{\frac{1}{2}}\|_F = \sqrt{\mathrm{tr}(K)}$ due to the coupling between the eigenvectors of the original matrix K and the perturbed eigenvectors as well as perturbed eigenvalues; and the third term proportional to ε^2 is due to the perturbed eigenvectors and perturbed eigenvalues. As expected, $||P - R\hat{P}||_F \to 0$ as $\varepsilon \to 0$, relating the LSD to the true factor with a rotation, which is consistent with Theorem 3. Our proof uses standard results from perturbation theory (Stewart, 1998).

Let $\hat{K} = \hat{P}^T\hat{P}$. By the triangle inequality,

$$\begin{aligned} ||K - \hat{K}||_F &= ||K - \tilde{K} + \tilde{K} - \hat{K}||_F \\ &\leq ||K - \tilde{K}||_F + ||\tilde{K} - \hat{K}||_F. \end{aligned}$$

By definition $\hat{K}$ minimizes $||K' - \tilde{K}||_F$ over all K' such that $K' = P'^T P', P' \geq 0, P^T \mathbf{1}_k = \mathbf{1}_n$. Therefore $||\hat{K} - \tilde{K}||_F \leq ||K - \tilde{K}||_F$ which yields,

$$\begin{aligned} ||K - \hat{P}^T\hat{P}||_F &= ||K - \hat{K}||_F \\ &\leq 2||K - \tilde{K}||_F \\ &= 2||W||_F \\ &\leq 2\varepsilon. \end{aligned}$$

Let $K = E\Lambda E^T$ and $\tilde{K} = \tilde{E}\tilde{\Lambda}\tilde{E}^T$ be the eigendecompositions of K and $\tilde{K}$ respectively. To get a bound on $\hat{P}$, we use a result by Stewart (1998) that states that

$$\tilde{E}_k = R_E E_k + W_E, \tag{23}$$

where $E_k, \tilde{E}_k$ are the first k columns of $E, \tilde{E}$ respectively, R_E is an orthogonal matrix and W_E is a $k \times n$ matrix that describes the perturbation of the eigenvectors. The $2-$norm of the perturbation matrix is bounded by the 2-norm of the additive noise as

$$||W_E||_2 \leq C_1 \frac{||W||_2}{|\delta_k|},$$

where $\delta_k = \tilde{\lambda}_k - \lambda_{k+1}$ is the difference between the k^{th} eigenvalue of the perturbed matrix $\tilde{K}$ and $(k+1)^{th}$ eigenvalue of the original matrix K.

Now, from (23), pre-multiplying $\tilde{E}_k^T$ by $\tilde{\Lambda}^{\frac{1}{2}}$ and post-multiplying by $R_1 = R_E^T R_0$ gives

$$\tilde{\Lambda}^{\frac{1}{2}}\tilde{E}_k^T R_1 = \tilde{\Lambda}^{\frac{1}{2}} E_k^T R_0 + \tilde{\Lambda}^{\frac{1}{2}} W_E^T R_1, \tag{24}$$

where R_0 is the rotation matrix that gives $P = \Lambda^{\frac{1}{2}} E_k R_0$. The matrix R_1 can be written as $\tilde{R}R$ such that $\hat{P} = \tilde{\Lambda}^{\frac{1}{2}}\tilde{E}_k\tilde{R}$. Using this decomposition and perturbation expansion of $\tilde{\Lambda}^{\frac{1}{2}} = \Lambda^{\frac{1}{2}} + W_\Lambda$, where by Mirsky's theorem (Stewart, 1998), $||W_\Lambda||_F \leq ||W||_F$, write (24) as

$$\begin{aligned} \hat{P}R &= (\Lambda^{\frac{1}{2}} + W_\Lambda)E_k^T R_0 + \tilde{\Lambda}^{\frac{1}{2}} W_E^T R_1, \\ &= P + W_\Lambda E_k^T R_0 + \tilde{\Lambda}^{\frac{1}{2}} W_E^T R_1. \end{aligned} \tag{25}$$

Then from (25),

$$
\begin{aligned}
||P-\hat{P}R||_F &= ||W_\Lambda E_k^T R_0 + \tilde{\Lambda}^{\frac{1}{2}} W_E^T R_1||_F \\
&\leq ||W_\Lambda E_k^T R_0||_F + ||\tilde{\Lambda}^{\frac{1}{2}} W_E^T R_1||_F \quad \text{(by triangle inequality)} \\
&= ||W_\Lambda||_F + ||\tilde{\Lambda}^{\frac{1}{2}} W_E||_F \quad \text{(by unitary-invariance of Frobenius norm)} \\
&\leq ||W||_F + ||\tilde{\Lambda}^{\frac{1}{2}} W_E||_F \quad \text{(by Mirsky's Theorem)} \\
&\leq ||W||_F + ||\tilde{\Lambda}^{\frac{1}{2}}||_F ||W_E||_F \quad \text{(by Cauchy Schwartz)} \\
&\leq ||W||_F + ||\Lambda^{\frac{1}{2}} + W_\Lambda||_F \; ||W_E||_F \quad \text{(perturbation expansion of eigenvalues)} \\
&\leq ||W||_F + \left(||\Lambda^{\frac{1}{2}}||_F + ||W||_F \right) ||W_E||_F \quad \text{(by triangle inequaity)} \\
&\leq \varepsilon + \left(||\Lambda^{\frac{1}{2}}||_F + \varepsilon \right) ||W_E||_F \quad \text{(from the bound on noise)} \\
&= \varepsilon + \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) ||W_E||_F \quad \text{(by definition)} \\
&\leq \varepsilon + \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) \sqrt{k}\; ||W_E||_2 \quad \text{(since rank of } W_E \text{ is at most } k\text{)} \\
&\leq \varepsilon + \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) \sqrt{k}\, C_1 \frac{||W||_2}{|\delta_k|} \quad \text{(from Stewart's perturbation theorem)} \\
&\leq \varepsilon + \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) \sqrt{k}\, C_1 \frac{||W||_F}{|\delta_k|} \quad \text{(since } \|W\|_2 \leq \|W\|_F\text{)} \\
&\leq \varepsilon + \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) \sqrt{k}\, C_1 \frac{\varepsilon}{|\delta_k|} \quad \text{(bound given on additive noise)} \\
&= \varepsilon \left(1 + C_1 \frac{\sqrt{k}}{|\delta_k|} \left(||K^{\frac{1}{2}}||_F + \varepsilon \right) \right),
\end{aligned}
$$

which gives the desired bound. ■

References

R. Arora. *Group Theoretical Methods in Signal Processing: Learning Similarities, Transformations and Invariants*. PhD thesis, Univ. of Wisconsin-Madison, 2009a.

R. Arora. On learning rotations. *Advances in Neural Information Processing Systems (NIPS)*, 2009b.

R. Arora and K. Livescu. Kernel CCA for multi-view learning of acoustic features using articulatory measurements. In *Symp. on Machine Learning in Speech and Language Processing*, 2012.

R. Arora and K. Livescu. Multi-view CCA-based acoustic features for phonetic recognition across speakers and domains. In *Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013.

R. Arora and W. A. Sethares. An efficient and stable algorithm for learning rotations. In *Proc. Intl. Conf. Pattern Recognition*, 2010.

R. Arora, M. R. Gupta, A. Kapila, and M. Fazel. Clustering by left-stochastic matrix factorization. *Proc. Intl. Conf. Machine Learning (ICML)*, 2011.

R. Arora, A. Cotter, K. Livescu, and N. Srebro. Stochastic optimization for PCA and PLS. In *50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

A. Berman and N. Shaked-Monderer. *Completely Positive Matrices*. World Scientific, 2003.

M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155 – 173, 2007.

M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. *Proc. Intl. Conf. Machine Learning (ICML)*, 2008.

M. Brown and S. Susstrunk. Multispectral SIFT for scene category recognition. *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

J. Bruna and S. Mallat. Classification with scattering operators. *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

L. Cazzanti, M. R. Gupta, and S. Srivastava. Fusing similarities and Euclidean features with generative classifiers. In *Proc. IEEE Conf. on Information Fusion (FUSION)*, 2009.

Y. Chen, E. K. Garcia, M. R. Gupta, L. Cazzanti, and A. Rahimi. Similarity-based classification: Concepts and algorithms. *JMLR*, 2009a.

Y. Chen, M. R. Gupta, and B. Recht. Learning kernels from indefinite similarities. In *Proc. Intl. Conf. Machine Learning (ICML)*, 2009b.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel target alignment. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

P. Deuflhard and M. Weber. Robust Perron cluster analysis in conformation dynamics. *Linear Algebra and Its Applications*, 398:161–184, 2005.

C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. *SIAM Conf. Data Mining*, 2005.

C. Ding, T. Li, and M. I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Trans. PAMI*, 32, 2010.

J. E. Driskell and T. McDonald. Identification of incomplete networks. Technical Report of Florida Maxima Corporation, 2008.

C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On using class-labels in evaluation of clusterings. *Proc. ACM SIGKDD*, 2010.

S. Feng, H. Krim, and I. A. Kogan. 3D face recognition using Euclidean integral invariants signature. *Proc. IEEE Workshop Statistical Signal Processing*, 2007.

E. Hanusa, D. Krout, and M. R. Gupta. Clutter rejection by clustering likelihood-based similarities. In *Proc. IEEE Conf. on Information Fusion (FUSION)*, 2011.

T. Hastie, R. Tibshirani, and J.Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2nd edition, 2009.

N.-D. Ho. *Nonnegative Matrix Factorizations Algorithms and Applications*. PhD thesis, Universite Catholique de Louvain, 2008.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *JMLR*, 5:1457–1469, 2004.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data*. Wiley Series in Applied Probability and Statistics. Wiley, 1990.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proc. of the Pacific Symposium on Biocomputing*, 2004.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems (NIPS)*, 2000.

Y. Li and A. Ngom. NMF Toolbox Ver. 1.0. `cs.uwindsor.ca/~li11112c/nmf.html`, 2011.

P. MacNaughton-Smith, W.T. Williams, M. B. Dale, and L. G. Mockett. Dissimilarity analysis: A new technique of hierarchical sub-division. *Nature*, 202:1034–5, 1964.

C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of R^n. *J. Opt. Theory Appl.*, 50:195–200, 1986.

J. K. Nelson and M. R. Gupta. An EM technique for multiple transmitter localization. *41st Conference on Information Science and Systems*, pages 610–615, 2007.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 2002.

D. Niu, J. G. Dy, and M. I. Jordan. Multiple non-redundant spectral clustering views. *Proc. Intl. Conf. Machine Learning (ICML)*, 2010.

P. Paatero. Least-squares formulation of robust non-negative factor analysis. *Chemometrics and Intell. Lab. Sys.*, 37:23 – 35, 1997.

P. Paatero. The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *J. Comp. Graph. Stat.*, 8(4):854–888, 1999.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

S. Philips, J. Pitton, and L. Atlas. Perceptual feature identification for active sonar echoes. *Proc. IEEE OCEANS*, 2006.

S. Selim and M. A. Ismail. K-means type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Trans. PAMI*, 6(1):81–87, 1984.

G. W. Stewart. *Matrix Algorithms, Volume I: Basic Decompositions*. SIAM, 1998.

U. von Luxburg. A tutorial on spectral clustering. Technical Report TR-149, Max Planck Institute for Biological Cybernetics, August 2006.

M. Weber and S. Kube. Robust Perron cluster analysis for various applications in computational life science. *Lecture Notes in Computer Science*, 3695:55–66, 2005.

M. Weber, W. Rungsarityotin, and A. Schliep. Perron cluster analysis and its connection to graph partitioning for noisy data. *Zentrum Informationstechnik Berlin Report*, 04-39, 2004.

H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. *Advances in Neural Information Processing Systems (NIPS)*, 2001.