Clustering by Left-Stochastic Matrix Factorization

Raman Arora RMNARORA@U.WASHINGTON.EDU
Maya R. Gupta GUPTA@EE.WASHINGTON.EDU
Amol Kapila AKAPILA@U.WASHINGTON.EDU
Maryam Fazel MFAZEL@U.WASHINGTON.EDU
University of Washington, Seattle, WA 98103, USA

Abstract

We propose clustering samples given their pairwise similarities by factorizing the similarity matrix into the product of a cluster probability matrix and its transpose. We propose a rotation-based algorithm to compute this left-stochastic decomposition (LSD). Theoretical results link the LSD clustering method to a soft kernel k-means clustering, give conditions for when the factorization and clustering are unique, and provide error bounds. Experimental results on simulated and real similarity datasets show that the proposed method reliably provides accurate clusterings.

1. Introduction

We propose a new non-negative matrix factorization (NNMF) model that yields a probabilistic clustering of samples given a matrix of pairwise similarities between the samples. The interpretation of non-negative factors of matrices as describing different parts of data was first given by (Paatero & Tapper, 1994) and (Lee & Seung, 1999). We discuss more related work in clustering by matrix factorization in Section 1.1. Then we introduce the proposed left-stochastic decomposition (LSD) clustering formulation in Section 1.2. We provide a theoretical foundation for LSD in Section 2. We exploit the geometric structure present in the problem to provide a rotation-based algorithm in Section 3. Experimental results are presented in Section 4.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

1.1. Related Work in Matrix Factorization

Some clustering objective functions can be written as matrix factorization objectives. Let n feature vectors be gathered into a feature-vector matrix $X \in \mathbb{R}^{d\times n}$. Consider the model $X \approx FG^T$, where $F \in \mathbb{R}^{d\times k}$ can be interpreted as a matrix with k cluster prototypes as its columns, and $G \in \mathbb{R}^{n\times k}$ is all zeros except for one (appropriately scaled) positive entry per row that indicates the nearest cluster prototype. The k-means clustering objective follows this model with squared error, and can be expressed as (Ding et al., 2005):

$$\arg \min_{\substack{F, G^TG=I \\ G\geq 0}} \|X - FG^T\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, and inequality $G \geq 0$ is component-wise. This follows because the combined constraints $G \geq 0$ and $G^TG = I$ force each row of G to have only one positive element. It is straightforward to show that the minimizer of (1) occurs at $F^* = XG$, so that (1) is equivalent to (Ding et al., 2005; Li & Ding, 2006):

$$\arg \min_{\substack{G^TG=I \\ G\geq 0}} \|X^TX - GG^T\|_F^2. \tag{2}$$

Replacing X^TX in (2) with a kernel matrix K results in the same minimizer as the kernel k-means objective (Ding et al., 2005; Li & Ding, 2006):

$$\arg \min_{\substack{G^TG=I \\ G\geq 0}} \|K - GG^T\|_F^2. \tag{3}$$

It has been shown that normalized-cut spectral clustering (Shi & Malik, 2000) also attempts to minimize an equivalent objective as kernel k-means, but for a kernel that is a normalized version of the input kernel (Ding et al., 2005). Similarly, probabilistic latent semantic indexing (PLSI) (Hofmann, 1999) can be formulated using the matrix factorization model $X \approx FG^T$, where the approximation is in terms of relative entropy (Li & Ding, 2006).

Ding et al. (2010) explored computationally simpler variants of the kernel k-means objective by removing the problematic constraint in (3) that $G^TG = I$, with the hope that the solutions would still have one dominant entry per row of G, which they take as the indicator of cluster membership. One such variant, termed *convex NMF*, restricts the columns of F (the cluster centroids) to be convex combinations of the columns of X. Another variant, *cluster NMF*, solves $\arg\min_{G\geq 0} \|X - XGG^T\|_F^2$.

1.2. LSD Clustering

We propose explicitly factorizing a matrix $K \in \mathbb{R}^{n\times n}$ to produce probabilities that each of the n samples belongs to each of k clusters. We require only that K be symmetric and that its top k eigenvalues be positive; we refer to such matrices in this paper as *similarity matrices*. Note that a similarity matrix need not be PSD – it can be an indefinite kernel matrix (Chen et al., 2009). If the input does not satisfy these requirements, it must be modified before use with this method. For example, if the input is a graph adjacency matrix A with zero diagonal, one could replace the diagonal by each node's degree, or replace A by $A + \alpha I$ for some constant α and the identity matrix I, or perform some other spectral modification. See (Chen et al., 2009) for further discussion of such modifications.

Given a similarity matrix K, we estimate a scaling factor c^*, and a cluster probability matrix P^* that solves

$$\arg\min_{c\in\mathbb{R}_+} \min_{\substack{P\geq 0\\ P^T\mathbf{1}_k=\mathbf{1}_n}} \|cK - P^TP\|_F^2, \tag{4}$$

where $\mathbf{1}_k$ is a $k \times 1$ vector of all ones. Note that a solution P^* to the above optimization problem is an approximate left-stochastic decomposition of the matrix c^*K, which inspires the following terminology:

LSD Clustering: A solution P^* to (4) is a cluster probability matrix, from which we form an *LSD clustering* by assigning the i^{th} sample to cluster j^* if $P_{j^*i} \geq P_{ji}$ for all $j \neq j^*$.

LSDable: We call K *LSDable* if the minimum of (4) is zero.

We show in Proposition 2 that the minimizing scaling factor c^* is given in a closed form and does not depend on a particular solution P^* for an LSDable K. Therefore, without loss of generality we will assume that $c^* = 1$ for an LSDable K.

1.3. Related Kernel Definitions

Some kernel definitions are related to our idealization that a given similarity matrix is usefully modeled as $cK \approx P^TP$. Cristianini et al. (2001) defined the *ideal kernel* K^* to be $K_{ij} = 1$ if and only if X_i and X_j are from the same class. The *Fisher kernel* (Jaakkola & Haussler, 1998) is defined as $K_{ij} = U_i^T\mathcal{I}^{-1}U_j$, where $\mathcal{I}$ is the Fisher information and each U_i is a function of the Fisher score, computed from a generative model and data X_i and X_j.

2. Theoretical Results

To provide intuition for the LSD clustering method and our rotation-based algorithmic approach, we present some theoretical results.

The proposed LSD clustering objective (4) is the same as the objective (3), except that we constrain P to be *left-stochastic*, factorize a scaled version of K, and do not constrain $P^TP = I$. The LSD clustering can be interpreted as a soft kernel k-means clustering, as follows:

Proposition 1 (Soft Kernel K-means) *Suppose K is* LSDable, *and let $K_{ij} = \phi(X_i)^T\phi(X_j)$ for some mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d_1}$ and some feature-vector matrix $X = [X_1, \ldots, X_n] \in \mathbb{R}^{d\times n}$, then the minimizer P^* of the LSD objective (4) also minimizes the following soft kernel k-means objective:*

$$\arg\min_{\substack{F\in\mathbb{R}^{d_1\times k}\\ P\geq 0, P^T\mathbf{1}_k=\mathbf{1}_n}} \|\Phi(X) - FP\|_F^2, \tag{5}$$

where $\Phi(X) = [\phi(X_1), \phi(X_2), \ldots, \phi(X_n)] \in \mathbb{R}^{d_1\times n}$.

Next, Theorem 1 states that there will be multiple left-stochastic factors P, which are related by rotations about the normal to the probability simplex (which includes permuting the rows, that is, changing the cluster labels):

Theorem 1 (Factors of K Related by Rotation) *Suppose K is* LSDable *such that $K = P^TP$. Then,*
(a) *for any matrix $M \in \mathbb{R}^{k\times n}$ s.t. $K = M^TM$, 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. $\hat{P} = R_uP$ and $R_u\vec{u} = \vec{u}$, where $\vec{u} = \frac{1}{k}[1, \ldots, 1]^T$ is normal to the plane containing the probability simplex.*
(c) *a sufficient condition for P to be unique (up to a permutation of rows), is that there exists an $m \times m$ sub-matrix in K which is a permutation of the identity matrix I_m, where $m \geq (\lfloor\frac{k}{2}\rfloor + 1)$.*

While there may be many LSDs of a given K, they may all result in the same LSD clustering. The following theorem states that for $k = 2$, the LSD clustering is unique, and for $k = 3$, it gives tight conditions for uniqueness of the LSD clustering. The key idea of this theorem for $k = 3$ is illustrated in Fig. 1.

Theorem 2 (Uniqueness of an LSD Clustering) *For $k = 2$, the LSD clustering is unique (up to a permutation of the labels). For $k = 3$, let α_c (α_{ac}) be the maximum angle of clockwise (anti-clockwise) rotation about 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 $\alpha_c < \beta_c$ or if $\alpha_{ac} < \beta_{ac}$, the LSD clustering is unique, up to a permutation of labels.*

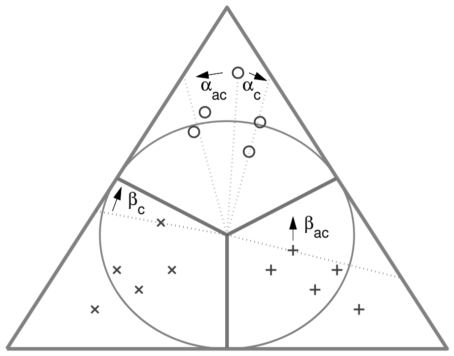


Figure 1. Illustration of conditions for uniqueness of the LSD clustering. The columns of P^* are shown as points on the probability simplex for $k = 3$ and $n = 15$. The points correspond to three clusters, marked by 'o's, '+'s, and 'x's. The Y-shaped thick gray lines show the cluster decision regions. Rotating the columns of P^* about the center of the simplex $\vec{u}$ results in a different LSD solution $\hat{P} = R_u P$ such that $K = \hat{P}^T\hat{P}$. One can rotate clockwise by at most angle β_c (and counter-clockwise β_{ac}) before any point crosses a cluster decision boundary. 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. As specified in Theorem 2, a sufficient condition for the LSD clustering to be unique (up to re-labeling) is if $\alpha_c < \beta_c$ and $\alpha_{ac} < \beta_{ac}$.

Our next result uses perturbation theory (Stewart, 1998) to give an error bound on LSD estimation for a perturbed LSDable matrix.

Theorem 3 (Perturbation Error Bound) *Suppose K is* LSDable *such that $K = P^T P$, where $P \in [0,1]^{k\times n}$. Let $\tilde{K} = K + W$, where W is a symmetric perturbation matrix with bounded Frobenius norm, $\|W\|_F \le \epsilon$. Then $\|K - \hat{P}^T\hat{P}\|_F \le 2\epsilon$, where $\hat{P}$ minimizes (4) 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*

$$\|P - R\hat{P}\|_F \le \epsilon \left(1 + C_1 \frac{\sqrt{k}}{|\tilde{\lambda}_k|}\left(\|K^{\frac{1}{2}}\|_F + \epsilon\right)\right), \quad (6)$$

where R is an orthogonal matrix and C_1 is a constant.

The error bound in (6) involves three terms: the first term captures the perturbation of the eigenvalues and scales linearly with ϵ; the second term involves $\|K^{\frac{1}{2}}\|_F = \sqrt{\text{tr}(K)}$ 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 $\epsilon \to 0$, relating the LSD to the true factor with a rotation, which is consistent with Theorem 1.

Proposition 2 (Scaling Factor) Given a similarity matrix K, the scaling factor c^* that solves

$$\arg\min_{c\in\mathbb{R}_+} \min_{\substack{P\in[0,1]^{k\times n}\\ P^T\mathbf{1}_k=\mathbf{1}_n}} \|cK - P^T P\|_F^2$$

is given by $c^* = \frac{\|(MM^T)^{-1}M\mathbf{1}_n\|_2^2}{k}$, where $K = M^T M$ is any decomposition of K.

3. Computing the LSD of K

The LSD problem stated in (4) is a nonconvex optimization in P and a *completely positive* (CP) factorization with an additional left-stochasticity constraint. CP problems are a subset of non-negative matrix factorization (NNMF) problems. Algorithms to solve NNMF can be applied here, such as Seidenberg's exponential-time quantifier elimination approach (Cohen & Rothblum, 1993); a multiplicative weights update (Lee & Seung, 2000); a greedy method using rank-one downdates (Biggs et al., 2008), or gradient descent (Paatero, 1999; Hoyer, 2004; Berry et al., 2007). The LSD problem could also be re-formulated and an alternating minimization approach used, as in (Paatero & Tapper, 1994). As is typical with nonconvex problems, no known polynomial-time method provides theoretical guarantees or conditions under which a correct solution is found.

We propose a rotation-based algorithm to solve (4) that exploits the geometry of this problem. We believe this is a new approach that may also be applicable to more general CP and NNMF problems. First, to build the reader's intuition, we present an overview of the algorithm solving (4). Full algorithmic details are deferred to Section 3.2.

3.1. Algorithm Overview

We overview the algorithm steps for an LSDable K. Full algorithm details, including projections needed if K is not LSDable, are given in Section 3.2. See Fig. 2 for a pictorial description of these steps for $k = 3$.

Step 1: Scale the similarity matrix K by c^* given in Proposition 2 to produce $K' = c^* K$.

Step 2: Factor $K' = M^T M$, for some $M \in \mathbb{R}^{k \times n}$.

Step 3: Rotate M to form $Q \in \mathbb{R}^{k \times n}$, whose column vectors all lie in the hyperplane that contains the probability simplex (see Fig. 2(a)).

Step 4: Rotate Q about the vector $\vec{u} = \frac{1}{k}[1, \ldots, 1]^T$ to form the left-stochastic solution P^*, whose column vectors all lie in the probability simplex (see Fig. 2(b)).

3.2. Full LSD Algorithm

Here we give more explicit details for each of the algorithm steps given in Sec. 3.1.

Step 1: Scale the similarity matrix K by c^* given in Proposition 2 to produce $K' = c^* K$.

Step 2a: Let $\Lambda_{1:k} \in \mathbb{R}^{k \times k}$ be the diagonal matrix of the top k eigenvalues of K' and $V_{1:k} \in \mathbb{R}^{n \times k}$ be the corresponding eigenvectors.

Step 2b: Set $M = \Lambda_{1:k}^{\frac{1}{2}} V_{1:k}^T$.

Step 2c: Compute vector $\vec{m} = (MM^T)^{-1} M \mathbf{1}_n$, which is normal to the least-squares hyperplane fit to the n columns of M obtained by solving for $\vec{m} = \arg\min_{\hat{m} \in \mathbb{R}^k} \|M^T \hat{m} - \mathbf{1}_n\|_2^2$.

Step 2d: Project the columns of M onto the hyperplane normal to $\vec{m}$ that is $1/\sqrt{k}$ units away; i.e., for the j^{th} column compute $\tilde{M}_j = M_j + \left(\vec{m}^T M_j - \frac{1}{\sqrt{k}}\right) \frac{\vec{m}}{\|\vec{m}\|}$. Let $\tilde{M} = [\tilde{M}_1, \ldots, \tilde{M}_n]$.

Step 3a: Compute a $k \times k$ rotation matrix R such that $R \frac{\vec{m}}{\|\vec{m}\|} = \frac{\vec{u}}{\|\vec{u}\|}$, where $\vec{u} = \frac{1}{k}[1, 1, \ldots, 1]^T$, as follows: Define $\vec{v} = \frac{\vec{u}}{\|\vec{u}\|} - \left(\frac{\vec{u}^T \vec{m}}{\|\vec{u}\| \, \|\vec{m}\|}\right) \frac{\vec{m}}{\|\vec{m}\|}$. Extend $\left(\frac{\vec{m}}{\|\vec{m}\|}, \frac{\vec{v}}{\|\vec{v}\|}\right)$ to a basis U for $\mathbb{R}^k$, using Gram-Schmidt orthogonalization. Define a $k \times k$ Givens rotation matrix R_G such that $(R_G)_{11} = (R_G)_{22} = \left(\frac{\vec{u}^T \vec{m}}{\|\vec{u}\| \, \|\vec{m}\|}\right)$, $(R_G)_{21} = -(R_G)_{12} = \left(\frac{\vec{u}^T \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}\right)$ and for all $i, j > 2$, set $(R_G)_{ij} = 1$ if $i = j$ and zero otherwise. Then $R = U R_G U^T$ is a $k \times k$ rotation matrix such that $R \frac{\vec{m}}{\|\vec{m}\|} = \frac{\vec{u}}{\|\vec{u}\|}$.

Step 3b: Set $Q = R\tilde{M}$.

Step 4a: Find a rotation R_u^* about $\vec{u}$ such that the projection $\lfloor R_u^* Q \rfloor$ of $R_u^* Q$ onto the probability simplex minimizes the factorization error to K':

$$R_u^* = \arg \min_{R : R\vec{u} = \vec{u}} \|K' - \lfloor RQ \rfloor^T \lfloor RQ \rfloor\|_F^2, \qquad (7)$$

such that R is a rotation matrix.

Note that for $k = 2$, no rotation is needed, and $P^* = \lfloor Q \rfloor$. For $k = 3$ and $k = 4$, minimizing the objective in (7) reduces to a simple optimization over a set of parameters, as discussed in Sec. 3.3. For general $k > 4$, any global optimization could be used, but we use an incremental batch approach detailed in Section 3.4. Projection onto the probability simplex can be found using the algorithm given in (Michelot, 1986).

Step 4b: Output $P^* = \lfloor R_u^* Q \rfloor$.

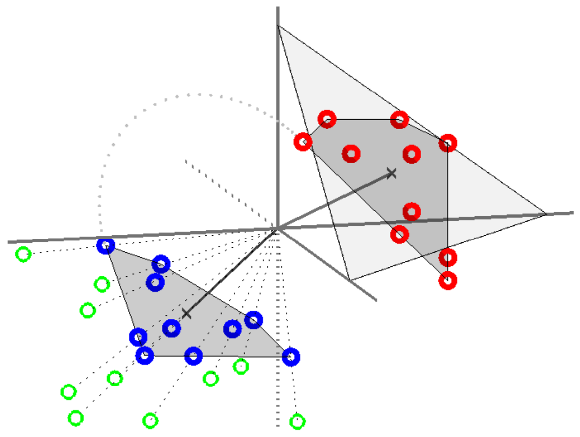
3.3. Further Details: Step 4a for $k = 3$ and $k = 4$

For $k \geq 3$, the rotation that leaves $\vec{u} = \frac{1}{k}[1, \ldots, 1]^T$ fixed can be described as the composition of three rotations: (a) a rotation R_{ue} that takes $\vec{u}$ to $\vec{e} = \frac{1}{\sqrt{k}}[0, 0, \ldots, 0, 1]^T$, followed by (b) a rotation R_e about $\vec{e}$, followed by (c) the rotation R_{ue}^T that takes $\vec{e}$ back to $\vec{u}$. In other words the rotation matrix can be decomposed as $R_u = R_{ue}^T R_e R_{ue}$, where R_{ue} is a fixed matrix and the optimization is over the rotation R_e. The rotation matrix R_e has the following structure:

$$R_e = \begin{bmatrix} R_{k-1} & 0 \\ 0^T & 1 \end{bmatrix}, \qquad (8)$$

where R_{k-1} is a $(k-1) \times (k-1)$ rotation matrix. This reduction in the dimensionality of the parameter space is due to the constraint $R\vec{u} = \vec{u}$. Thus for $k = 3$, the optimization is over planar rotation matrices that can be parameterized by the angle of rotation $\theta \in [0, 2\pi)$, thereby reducing the problem to a 1-D optimization problem.

For $k = 4$, the rotation matrix R_e is a 3×3 matrix that can be parameterized by ZYZ Euler angles α, β, γ; i.e., angles of rotation about Z, Y and Z axis, respectively:


(a) Rotate M to the simplex hyperplane

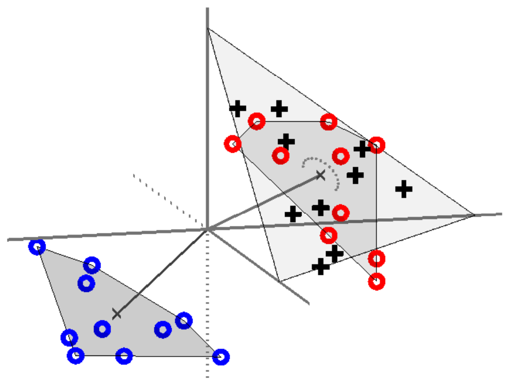

(b) Rotate to fit the simplex

Figure 2. The proposed rotation-based LSD algorithm for an example where $k = 3$. Fig. 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 Fig. 2b, we rotate the columns of Q about the normal $\vec{u}$ 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^*.

$R_e(\alpha, \beta, \gamma) = R_Z(\gamma) R_Y(\beta) R_Z(\alpha)$, where $\alpha, \gamma \in [0, 2\pi)$ and $\beta \in [0, \pi]$. Therefore, the optimization in (7) reduces to a search over three parameters.

3.4. Further Details: Step 4a for $k > 4$

For higher k, we use an incremental batch algorithm for learning rotations (Arora, 2009a;b) as follows: Let $J(R)$ denote the objective function in (7). Initialize the search with $R_u = I$ and $R_u^* = I$, $J^* = J(R_u^*)$. Repeat the following steps for a fixed number of passes or until all columns of $(R_u Q)$ lie inside the simplex.

1. Let $\mathcal{I}$ be the set of indices of columns of the matrix $(R_u Q)$ that lie outside the simplex. Initialize T as a $(k-1) \times (k-1)$ zero matrix.

2. For each $j \in \mathcal{I}$,

   (a) project the j^{th} column $(R_u Q)_j$ onto the simplex to get $\lfloor (R_u Q)_j \rfloor$ (Michelot, 1986),

   (b) rotate $(R_u Q)_j$ and $\lfloor (R_u Q)_j \rfloor$ about the origin to lie in the hyperplane normal to $\vec{e}$, and then take the first $k-1$ components of the rotated vectors:

   $$\begin{aligned} \vec{x}_j &= (R_{ue} (R_u Q)_j)_{1:k-1}, \\ \vec{y}_j &= (R_{ue} \lfloor (R_u Q)_j \rfloor)_{1:k-1}, \end{aligned}$$

   where $\vec{x}_j, \vec{y}_j \in \mathbb{R}^{k-1}$,

   (c) update $T = T + \vec{y}_j \, \vec{x}_j^T$.

3. Compute the singular value decomposition of $T = U \Sigma V^T$. Let $\tilde{\Sigma}$ be the diagonal matrix such that $\tilde{\Sigma}_{ii} = 1$ for $i = 1, \ldots, k-2$ and $\tilde{\Sigma}_{k-1,k-1} = \det(U)\det(V)$. Compute the rotation matrix $R_{k-1} = U \tilde{\Sigma} V^T$.

4. Form R_e using R_{k-1} as in (8), and update the rotation matrix $R_u = \left(R_{ue}^T R_e R_{ue} \right) R_u$.

5. If $J(R_u) < J^*$, update $R_u^* = R_u$ and $J^* = J(R_u^*)$.

The motivation for this update is that the $(k-1) \times (k-1)$ matrix R_{k-1} computed at each iteration solves:

$$\arg \min_{\substack{RR^T = I, R^T R = I \\ \det(R) = 1}} \sum_{j \in \mathcal{I}} \| \vec{y}_j - R \vec{x}_j \|_2^2 .$$

4. Experimental Results

We compare clustering algorithms that take a pairwise-similarity matrix K as input. We present two Gaussian simulations, two simulations where the clusters are defined by manifolds, and real similarity datasets from (Chen et al., 2009).

Every time the k-means algorithm is used, it is run with 20 random initializations (which are each given the Matlab default of 100 maximum iterations), and the result that minimizes the within-cluster scatter is used. K-means is implemented with Matlab's `kmeans` routine, except for the kernel k-means which we implemented. The *convex NMF* routine used NMFlib (Grindlay, 2008). For normalized spectral clustering, we used the Ng-Jordan-Weiss version (Ng et al., 2002).

4.1. Gaussian Cluster Simulations

We ran two Gaussian cluster simulations for $k = 2$ and $k = 3$ clusters. The first simulation generated samples iid from $k = 2$ Gaussians $\mathcal{N}([0\,0]^T, [0.5\sigma^2\,0; 0\,4\sigma^2])$ and $\mathcal{N}([5\,0]^T, [4\sigma^2\,0; 0\,0.5\sigma^2])$. Then the RBF kernel matrix with bandwidth one was computed. For each value of σ^2, we averaged results for 1000 randomly generated datasets, each of which consisted of 100 samples from each cluster.

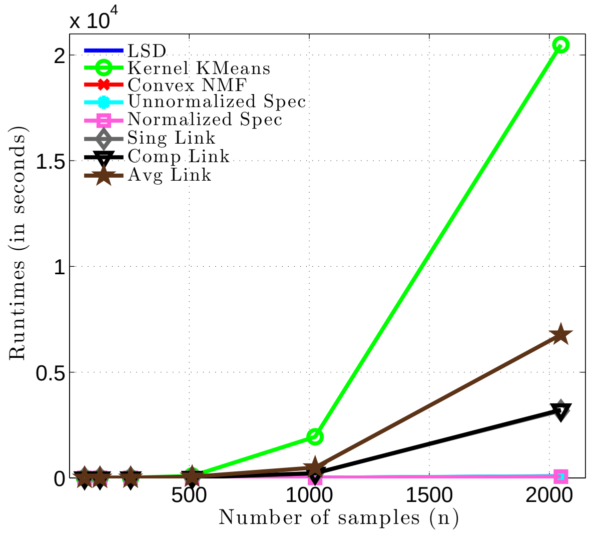
The second simulation used the same two Gaussian clusters with $\sigma^2 = 3$, plus a third cluster generated iid from $\mathcal{N}([2\,4]^T, [3\,0; 0\,3])$. We averaged results for 1000 randomly generated datasets, each of which consisted of 50 samples from each cluster.

In Fig. 3, we show runtimes for increasing n for the two-cluster Gaussian simulation. The spectral clusterings, convex NMF, and LSD clustering all have comparable runtimes, and are much faster than the linkages or kernel k-means. Theoretically, we hypothesize that the LSD clustering computational complexity is roughly the same as spectral clustering, as both will be dominated by the need to compute k eigenvectors.

Results for the two-cluster simulation are shown in Fig. 3. LSD shows similar performance to kernel k-means and convex NMF, with LSD performing slightly better for most values of σ^2. Results are similar for the three-cluster simulation, shown in Fig. 4.

4.2. Manifold Simulations

We simulate $k = 2$ clusters lying on two one-dimensional manifolds in two-dimensional space, as shown in Fig. 5. For each sample, the first feature $x[1]$ is drawn iid from a standard normal, and the second feature is $x[1]^2$ if the point is from the first cluster, or $\arctan(x[1]) - b$ for some $b \in \mathbb{R}$ if the point is from the second cluster. Fifty samples are randomly generated for each cluster, and results were averaged for 250 randomly generated datasets for each value of the cluster separation b. We clustered based on two kernels. The first kernel was the RBF kernel with bandwidth 1. The second kernel was a nearest-neighbor kernel $K = A + A^T$, where $A(x_i, x_j) = 1$ if x_i is one of the five nearest neighbors of x_j (includes itself as a neighbor) and $A(x_i, x_j) = 0$ otherwise.



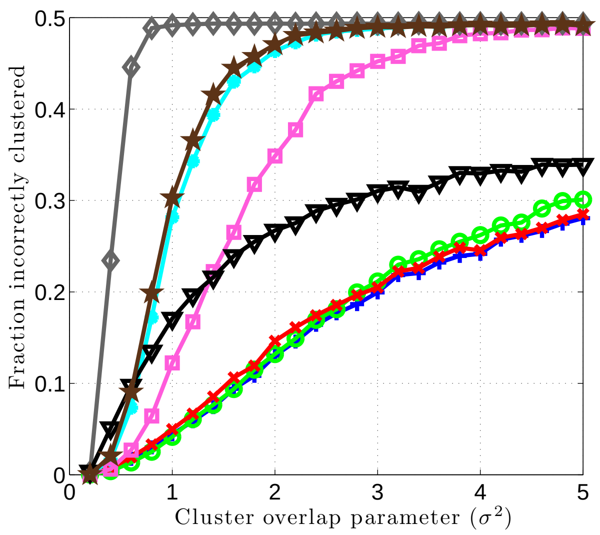


Figure 3. Runtimes (top) and results (bottom) for the two-cluster Gaussian simulation.

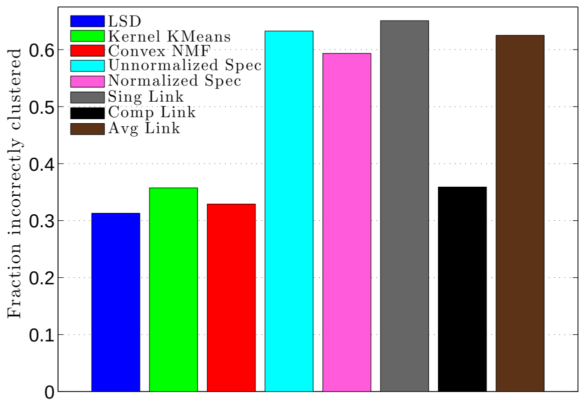


Figure 4. Results for three-cluster Gaussian simulation.

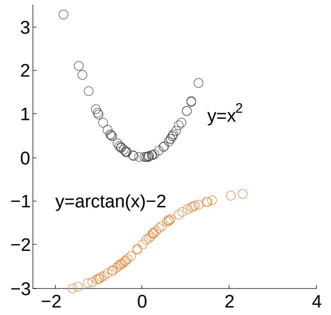
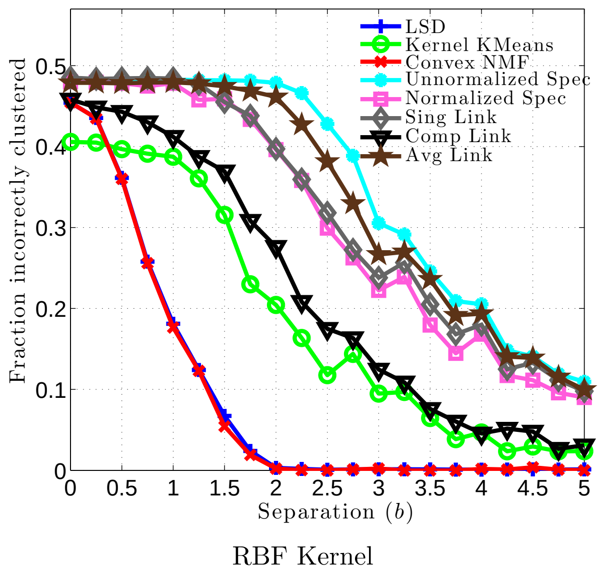


Figure 5. One realization of the $n = 100$ samples in the manifold simulation with separation $b = 2$.



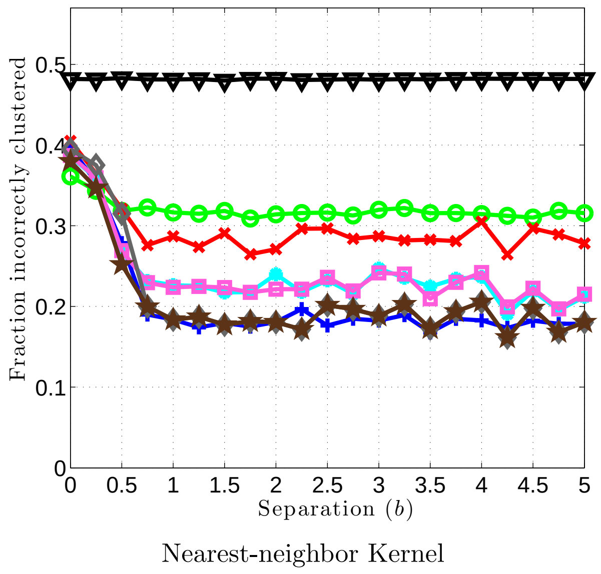


Figure 6. Results for the manifold simulation.

Results are shown in Fig. 6. Given the RBF kernel, LSD clustering and convex NMF outperform the other methods, followed by kernel k-means and complete linkage. Given the nearest-neighbor kernel, LSD clustering, single-linkage, and average-linkage perform best, followed by the spectral clusterings, with convex NMF and kernel k-means doing roughly 10% worse than LSD.

4.3. Real Similarity Data

We also compared the clustering methods on ten real similarity datasets used in (Chen et al., 2009); these datasets are provided as a pairwise matrix K, and the similarity definitions include human judgement, Smith-Waterman distance, and purchasing trends; see (Chen et al., 2009) for more details. These datasets have class labels for all samples, which were used as the ground truth to compute the errors shown in Table 4.3. While no one clustering method clearly outshines all of the others in Table 4.3, the proposed LSD clustering is ranked first for four of the ten datasets.

5. Discussion

In this paper, we proposed a left-stochastic decomposition of a similarity matrix to produce cluster probability estimates. Compared to seven other clustering methods given the same similarity matrix as input, the LSD clustering reliably performed well across the different simulations and real benchmark datasets.

To compute the LSD, we proposed a rotation-based algorithm for the matrix factorization that may be an effective approach to solving other NNMF and completely positive problems. Even though the algorithm uses only the top k eigenvectors, the perturbation error bound of Theorem 3 still holds because the assumptions on the perturbation matrix ensure that there is a correspondence between the top k eigenvectors of the original and perturbed matrices.

One notable advantage of the LSD clustering algorithm is that it is deterministic for $k = 2$; that is, the clustering only requires computing the top two eigenvectors, a rotation to the positive orthant, and projecting to the probability simplex. This is in contrast to methods like kernel k-means and spectral clustering, which are usually implemented with multiple random initializations of k-means. As a consequence, LSD clustering can provide a deterministic top-down binary clustering tree, a topic for future research. Overall, the runtime for LSD clustering can be expected to be similar to spectral clustering. For $k = 3$, we presented theoretical conditions for the uniqueness of an LSD clustering, which we believe may be satisfied in prac-

Table 1. Clustering Error Rate on Similarity Datasets

	Amazon Binary	Aural Sonar	Internet Ads	Patrol	Protein	Voting	Yeast Pfam 7-12	Yeast SW 5-7	Yeast SW 5-12	Yeast SW 7-12
# of clusters k:	*2*	*2*	*2*	*8*	*4*	*2*	*2*	*2*	*2*	*2*
LSD	.39	.14	.35	**.44**	.20	.10	.36	**.28**	**.09**	**.10**
Kernel K-Means	**.33**	.19	.29	.61	.28	.10	.49	.37	.10	.16
Convex NMF	**.33**	**.11**	.50	.79	.36	.10	.38	.28	.10	**.10**
Unnorm. Spec	.39	.49	**.16**	.57	.65	.09	.41	.50	.11	.50
Norm. Spec	.39	.13	.19	.79	**.15**	.10	.38	.31	.14	**.10**
Sing. Link	.39	.49	**.16**	.79	.66	.38	.50	.50	.50	.50
Comp. Link	.39	.47	**.16**	.68	.59	**.04**	**.34**	.39	.18	.33
Avg. Link	.39	.41	**.16**	.64	.62	.05	.47	.40	.10	.15

tice. We hypothesize that analogous conditions exist for higher k.

References

Arora, R. On learning rotations. *NIPS*, 2009a.

Arora, R. *Group theoretical methods in signal processing: learning similarities, transformations and invariants*. PhD thesis, Univ. of Wisconsin, 2009b.

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

Biggs, M., Ghodsi, A., and Vavasis, S. Nonnegative matrix factorization via rank-one downdate. *ICML*, 2008.

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

Cohen, J. E. and Rothblum, U. G. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and Its Applications*, 190: 149–168, 1993.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. On kernel target alignment. *NIPS*, 2001.

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

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

Grindlay, G. NMFlib. www.ee.columbia.edu/ grindlay, 2008.

Hofmann, T. Probabilistic latent semantic analysis. *UAI*, 1999.

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

Jaakkola, T. and Haussler, D. Exploiting generative models in discriminative classifiers. *NIPS*, 1998.

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

Lee, D. D. and Seung, H. S. Algorithms for non-negative matrix factorization. *NIPS*, 2000.

Li, T. and Ding, C. The relationships among various nonnegative matrix factorization methods for clustering. In *Proc. Intl. Conf. Data Mining*, 2006.

Michelot, C. 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.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *NIPS*, 2002.

Paatero, P. 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.

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

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8), 2000.

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