# SoF: Soft-Cluster Matrix Factorization for Probabilistic Clustering

**Han Zhao**[†], **Pascal Poupart**[†], **Yongfeng Zhang**[§] and **Martin Lysy**[‡]
[†]David R. Cheriton School of Computer Science, University of Waterloo, Canada
[§]Department of Computer Science and Technology, Tsinghua University, China
[‡]Department of Statistics and Actuarial Science, University of Waterloo, Canada
[†‡]{han.zhao, ppoupart, mlysy}@uwaterloo.ca, [§]zhangyf07@gmail.com

## Abstract

We propose SoF (Soft-cluster matrix Factorization), a probabilistic clustering algorithm which softly assigns each data point into clusters. Unlike model-based clustering algorithms, SoF does not make assumptions about the data density distribution. Instead, we take an axiomatic approach to define 4 properties that the probability of co-clustered pairs of points should satisfy. Based on the properties, SoF utilizes a distance measure between pairs of points to induce the conditional co-cluster probabilities. The objective function in our framework establishes an important connection between probabilistic clustering and constrained symmetric Nonnegative Matrix Factorization (NMF), hence providing a theoretical interpretation for NMF-based clustering algorithms. To optimize the objective, we derive a sequential minimization algorithm using a penalty method. Experimental results on both synthetic and real-world datasets show that SoF significantly outperforms previous NMF-based algorithms and that it is able to detect non-convex patterns as well as cluster boundaries.

## Introduction

Clustering algorithms have played an important role in various research tasks, including but not limited to document clustering (Steinbach et al. 2000; Xu, Liu, and Gong 2003), image segmentation (Shi and Malik 2000) and gene clustering (Thalamuthu et al. 2006). Recently, probabilistic clustering has gained in popularity and application (He et al. 2011; Jain 2010; Aggarwal and Reddy 2013) because the probabilistic nature of the clusters facilitates their integration into several tasks including latent semantic indexing (Ding, Li, and Peng 2008), probabilistic pre-training in deep learning techniques (Erhan et al. 2010), etc. Most of the existing probabilistic clustering algorithms attempt to infer the posterior cluster distribution by assuming a family of density distributions, typical examples include the Gaussian mixture model, the Dirichlet process (Biernacki, Celeux, and Govaert 2000), the Wishart-Dirichlet process (Vogt et al. 2010), etc. However, the assumed family may not include the distribution of the data, hence a flexible and data-driven approach is desirable.



In this paper, we propose a novel probabilistic clustering algorithm based on Soft-cluster Matrix Factorization (SoF). Different from previous approaches, SoF does not make assumptions about the data density distribution. Instead, SoF employs a distance function between pairs of input points to induce their co-cluster probabilities. By directly considering the co-cluster probabilities, SoF can model clusters with various patterns, including both convex and non-convex clusters. The input to SoF consists of a set of data points, a distance function $L$ over these data points, and the number of clusters, $K$. As we will see later, the choice of distance function $L$ reflects our prior knowledge about the clustering task.

SoF has a close connection with constrained symmetric NMF. In the past few years, algorithms for solving NMF have been studied extensively (Seung and Lee 2001; Kim and Park 2008a; 2008b; 2008c; Kuang, Park, and Ding 2012) and applied to clustering tasks. However, to the best of our knowledge, the connection between NMF and probabilistic clusterings is not well understood. The objective function in our framework reveals an intrinsic connection between probabilistic clustering and constrained symmetric NMF, hence giving a theoretical justification for the application of symmetric NMF techniques to clustering tasks. Furthermore, we will show that the true pairwise co-cluster matrix for probabilistic clustering is actually a *completely positive matrix*, which is NP-hard to determine (Berman and Shaked-Monderer 2003). Our clustering algorithm can then be viewed as approximating the completely positive matrix in a relaxed cone space, which is a better approximation than other existing NMF based algorithms. To solve SoF efficiently, we derive a sequential minimization framework to optimize the objective. Experiments on both synthetic and real-world datasets show that SoF is able to find non-convex clusters due to the absence of any density assumption and its probabilistic nature can be used to detect boundaries between multiple clusters.

## Preliminaries

Given a set of input points $I = \{\mathbf{v}_i \in \mathbb{R}^d \mid i = 1, \ldots, N\}$ and the number of clusters $K$, the task of probabilistic clustering is to associate a probability vector $\mathbf{p} \in \mathbb{R}^K$ to each point $\mathbf{v} \in I$, where the $j$th component of $\mathbf{p}$ represents the probability that $\mathbf{v}$ is assigned to the $j$th cluster. The proba-

bility vector $\mathbf{p}$ should satisfy $\mathbf{p} \succeq \mathbf{0}$ and $\mathbf{1}^T\mathbf{p} = 1$.[1]

Let $\mathbf{V}$ be a continuous random vector in $\mathbb{R}^d$ for input points and $C$ be a categorical random variable which takes a cluster label from $\{1, \ldots, K\}$. We use $p(\mathbf{V}, C)$ to denote the joint probability distribution over $domain(\mathbf{V}) \times domain(C)$. Each sample from the joint probability distribution is a pair $(\mathbf{v}, c)$, where $\mathbf{v}$ is the point we observe and $c$ denotes the cluster label associated with $\mathbf{v}$. We assume that all the pairs (point and its label) are sampled from a mixture model independently. More precisely, we assume there are $K$ components where each component corresponds to a cluster. Each component can generate observed data based on its conditional distribution $p(\mathbf{V}|C = k)$, $k = 1, \ldots, K$ independently. Unlike traditional mixture models, we do not assume any fixed form for the conditional distributions $p(\mathbf{V}|C = k)$. Moreover, for different $k$'s, the conditional distribution can take different forms.

## Soft-cluster Matrix Factorization

We formally define a probabilistic clustering function as follows:

**Definition 1.** Given an input set $I = \{\mathbf{v}_i \in \mathbb{R}^d \mid i = 1, \ldots, N\}$ with some fixed dimension $d$, a distance function $L : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ that is reflexive, symmetric and positive definite[2], and a fixed number of clusters $K$, a probabilistic clustering algorithm is a function that takes $I, L, K$ as inputs and for each $\mathbf{v} \in I$, outputs a vector $\mathbf{p}_\mathbf{v} \in \mathbb{S}^{K-1}$, where $\mathbb{S}^{K-1}$ represents the $K-1$ dimensional probability simplex.

Given $I, L, K$, a probabilistic clustering algorithm should, for each $\mathbf{v} \in I$, output a probability vector which is close to the true posterior probability vector, i.e., $\mathbf{p}_\mathbf{v} \approx (p(C = 1|\mathbf{V} = \mathbf{v}), \ldots, p(C = K|\mathbf{V} = \mathbf{v}))^T$. The choice of distance function $L(\cdot, \cdot)$ reflects our prior knowledge about the clustering task, where the basic assumption is that points which are close in terms of $L$ tend to reside in the same cluster and points which are far away from each other in terms of $L$ tend to reside in different clusters.

### Co-cluster Probability

The distance function $L : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ measures how close two points are. One of the key questions in clustering is how can we obtain a similarity measure between input points in terms of the distance function $L$? More specifically, what is the probability that two points belong to the same cluster?

We propose the following four basic properties to find appropriate expressions for the probability that a pair of points is co-clustered based on the distance function $L$. Hence, $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^d$, we will use $p_C(\mathbf{v}_1, \mathbf{v}_2)$ to denote the *empirical* probability that $\mathbf{v}_1$ and $\mathbf{v}_2$ reside in the same cluster. A co-cluster probability $p_C(\cdot, \cdot)$ should satisfy the following four properties:

1. Boundary property: $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^d$, if $L(\mathbf{v}_1, \mathbf{v}_2) = 0$, then $p_C(\mathbf{v}_1, \mathbf{v}_2) = 1$; if $L(\mathbf{v}_1, \mathbf{v}_2) \to \infty$, then $p_C(\mathbf{v}_1, \mathbf{v}_2) \to 0$.
2. Symmetry property: $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^d$, $p_C(\mathbf{v}_1, \mathbf{v}_2) = p_C(\mathbf{v}_2, \mathbf{v}_1)$.
3. Monotonicity property: $\forall \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^d$, $p_C(\mathbf{v}_1, \mathbf{v}_2) \leq p_C(\mathbf{v}_1, \mathbf{v}_3)$ if and only if $L(\mathbf{v}_1, \mathbf{v}_2) \geq L(\mathbf{v}_1, \mathbf{v}_3)$.
4. Tail property: Given $\forall \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^d$, $\left|\frac{\partial p_C(\mathbf{v}_1, \mathbf{v}_2)}{\partial L(\mathbf{v}_1, \mathbf{v}_2)}\right| \leq \left|\frac{\partial p_C(\mathbf{v}_1, \mathbf{v}_3)}{\partial L(\mathbf{v}_1, \mathbf{v}_3)}\right|$ if and only if $L(\mathbf{v}_1, \mathbf{v}_2) \geq L(\mathbf{v}_1, \mathbf{v}_3)$

The boundary property forces $p_C(\cdot, \cdot)$ to be a valid probability metric and the symmetry property allows the arguments of $p_C(\cdot, \cdot)$ to be interchangeable. The monotonicity property formalizes our intuition that closer points tend to be more likely to reside in the same cluster. This property also suggests that $L(\cdot, \cdot)$ should be well-chosen to reflect our prior knowledge about the data. The tail property can be interpreted as follows: changes in the distance of two points will have an impact on the co-cluster probability that diminishes with the increase of the distance between them.

**Proposition 1.** Given a distance function $L : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, there exists a family of co-cluster probability functions $p_C(\mathbf{v}_1, \mathbf{v}_2) = e^{-cL(\mathbf{v}_1, \mathbf{v}_2)}$ which satisfy the boundary, symmetry, monotonicity and tail properties simultaneously for any constant $c > 0$.

*Proof.* The boundary property is satisfied since $e^{-c0} = 1$ and $e^{-c\infty} = 0$. The symmetry property follows from the fact that $L$ is symmetric. The monotonicity property holds since $e^{-cL(\mathbf{v}_1, \mathbf{v}_2)} \leq e^{-cL(\mathbf{v}_1, \mathbf{v}_3)} \iff L(\mathbf{v}_1, \mathbf{v}_2) \geq L(\mathbf{v}_1, \mathbf{v}_3)$. The tail property can be derived as follows:

$$\begin{aligned}
&\left|\frac{\partial p_C(\mathbf{v}_1, \mathbf{v}_2)}{\partial L(\mathbf{v}_1, \mathbf{v}_2)}\right| \leq \left|\frac{\partial p_C(\mathbf{v}_1, \mathbf{v}_3)}{\partial L(\mathbf{v}_1, \mathbf{v}_3)}\right| \\
\iff &\left|-ce^{-cL(\mathbf{v}_1, \mathbf{v}_2)}\right| \leq \left|-ce^{-cL(\mathbf{v}_1, \mathbf{v}_3)}\right| \\
\iff &e^{-cL(\mathbf{v}_1, \mathbf{v}_2)} \leq e^{-cL(\mathbf{v}_1, \mathbf{v}_3)} \\
\iff &L(\mathbf{v}_1, \mathbf{v}_2) \geq L(\mathbf{v}_1, \mathbf{v}_3)
\end{aligned}$$

□

For the specific choice where $L$ is the squared Euclidean distance, our construction of the co-cluster probability function coincides with the Gaussian kernel. Similarly, when $L$ is the Euclidean distance, the co-cluster probability function is equivalent to the Laplacian kernel. Note however that our framework is more general in the sense that one may use other kinds of distance functions $L$ that may not lead to well-known kernels. In $e^{-cL(\mathbf{v}_1, \mathbf{v}_2)}$ we have a scaling parameter $c$. When $c \to 0$, the probability that any two points belong to the same cluster tends to 1 and a single global cluster holds all the data points; when $c \to \infty$, the co-cluster probability of any two points tends to 0 and each point forms its own cluster. Hence, the clustering quality is very sensitive to the choice of scaling parameter $c$. Later we will discuss a strategy to avoid manually tunning the scaling parameter $c$ while at the same time achieving the *scaling invariant* property of

[1] We use $\succeq$ to denote the element-wise greater than between matrices of the same dimension.

[2] The distance function $L(\cdot, \cdot)$ is not required to be a metric.

clustering (Kleinberg 2003). Henceforth, for simplicity of the notation, we will implicitly assume $c = 1$.

## Probabilistic Clustering

We use $P \in \mathbb{R}^{N\times N}$ to denote the soft-cluster matrix, where $P_{ij} \triangleq p_C(\mathbf{v}_i, \mathbf{v}_j) = e^{-L(\mathbf{v}_i,\mathbf{v}_j)}$ is the *empirical* probability that two points are clustered together. In our setting, $P$ acts as the refined prior knowledge extracted from the raw input to guide the probabilistic clustering algorithm in the learning process.

Based on our probabilistic model, given $\mathbf{v}_i, \mathbf{v}_j$, $i \neq j$, the *true* probability that $\mathbf{v}_i, \mathbf{v}_j$ belong to the same cluster, $\Pr(\mathbf{v}_i \sim \mathbf{v}_j)$, is given by

$$\begin{aligned}\Pr(\mathbf{v}_i \sim \mathbf{v}_j) &= \sum_{k=1}^{K} p(c_i = c_j = k|\mathbf{v}_i, \mathbf{v}_j) \\ &\overset{\zeta_1}{=} \sum_{k=1}^{K} \frac{p(\mathbf{v}_i, c_i = k)p(\mathbf{v}_j, c_j = k)}{p(\mathbf{v}_i, \mathbf{v}_j)} \\ &\overset{\zeta_2}{=} \sum_{k=1}^{K} p(c_i = k|\mathbf{v}_i)p(c_j = k|\mathbf{v}_j) = \mathbf{p}_{\mathbf{v_i}}^T \mathbf{p}_{\mathbf{v_j}}\end{aligned}$$

where $\zeta_1$ and $\zeta_2$ follow from the i.i.d. assumption of the data generation process.

Given the joint probability distribution $p(\mathbf{V}, C)$, we can compute the conditional probability distribution of cluster label $C$ given $\mathbf{V}$ by $p(C|\mathbf{V}) = \frac{p(\mathbf{V},C)}{p(\mathbf{V})}$. Let $W = [W_{ij}]_{N\times K} \in \mathbb{R}_+^{N\times K}$, where $W_{ij} = p(c_i = j|\mathbf{v}_i)$. We rewrite $\Pr(\mathbf{v}_i \sim \mathbf{v}_j)$ as:

$$\Pr(\mathbf{v}_i \sim \mathbf{v}_j) = \sum_{k=1}^{K} W_{ik}W_{jk} = W_{i:}W_{:j}^T \tag{1}$$

where $W_{i:}$ represents the $i$th row vector of matrix $W$. Let's define $\Pr(\mathbf{V} \sim \mathbf{V}') \triangleq [\Pr(\mathbf{v}_i \sim \mathbf{v}_j)]_{N\times N}$ and express it in matrix form:

$$\Pr(\mathbf{V} \sim \mathbf{V}') = WW^T \tag{2}$$

The $i$th row of $W$ corresponds to the conditional probability vector that $\mathbf{v}_i$ belongs to different clusters, i.e., $W_{i:} = \mathbf{p}_{\mathbf{v}_i}^T$. We see that the *true* co-cluster probability matrix $\Pr(\mathbf{V} \sim \mathbf{V}')$ is both nonnegative and symmetric positive semidefinite. Furthermore, it can be factorized as the product of a nonnegative matrix and its transpose. Such a matrix is known as the *Completely Positive matrix* (C.P. matrix) in the literature (Berman and Shaked-Monderer 2003; Dickinson and Gijben 2014). C.P. matrices are contained in the intersection of the positive semidefinite cone and the nonnegative cone, but they are a strict subset of this class (Dickinson and Gijben 2014). We will use the C.P. cone to denote the set of C.P. matrices. It has recently been proven that determining whether a given matrix is in the C.P. cone is NP-hard (Dickinson and Gijben 2014). From this perspective, our construction of the soft-cluster matrix is actually a relaxation that allows us to search in the intersection of the nonnegative cone and the positive semidefinite cone, which is a strict superset of the C.P. cone.

Our goal is to learn the matrix $W$ from the inputs $I, L, K$. Given $I, L, K$, the soft-cluster matrix $P$, which satisfies the four properties and admits the form $e^{-L(\cdot,\cdot)}$, is a relaxation and approximation of the true co-cluster matrix $\Pr(\mathbf{V} \sim \mathbf{V}')$. Therefore we obtain a natural learning paradigm to estimate the optimal $W$, denoted by $W^*$:

$$\begin{array}{ll}\text{minimize}_W & ||P - WW^T||_F^2 \\ \text{subject to} & W \in \mathbb{R}_+^{N\times K}, W\mathbf{1}_K = \mathbf{1}_N\end{array} \tag{3}$$

where $\mathbf{1}_K$ and $\mathbf{1}_N$ are $K$ and $N$ dimensional column vectors whose components are all 1; $||\cdot||_F$ represents the Frobenius norm of a matrix where $||A||_F = (\sum_{i,j} A_{ij}^2)^{1/2}$. Before tackling the optimization problem above, we discuss some of its properties and its relations to other clustering algorithms.

1) SoF is closely related to other clustering algorithms, including Kernel Kmeans (Dhillon, Guan, and Kulis 2004), Spectral Clustering (Ding, He, and Simon 2005) and Symmetric Nonnegative Matrix Factorization (SymNMF) (Kuang, Park, and Ding 2012), but it also exhibits significant differences. We list the connections and differences among those methods in Table 1. In summary, the matrix $P$ used in SoF is nonnegative as well as symmetric positive semidefinite (S.P.D.), which distinguishes it from all other three methods. Meanwhile, the linear constraints $W\mathbf{1} = \mathbf{1}$ together with the nonnegativity constraint in SoF make the solution clearly interpretable as a probabilistic assignment of points to clusters, while all other three methods seek to find data representations in either kernel spaces or reduced eigenspaces.

2) The decision version of the optimization problem in (3) is closely connected to the strong membership problem for the C.P. cone, which asks given an S.P.D matrix $P$ which is nonnegative, whether there exists a nonnegative matrix $W$ such that $P = WW^T$. It's then easily seen that the decision version of (3) is essentially a restricted version of the strong membership problem for the C.P. cone, which is NP-hard. We conjecture that the decision version of (3) is also NP-hard because of its close connection to the strong membership problem for the C.P. cone. If this is the case, the optimization problem formulated in (3) is also NP-hard. However, the validation of this conjecture is out of the scope of this paper and we leave it for future work.

3) The optimization problem (3) is non-convex in general because the objective function contains a fourth order polynomial in $W$ whose Hessian matrix cannot be guaranteed to be positive semi-definite. The nonnegative constraint together with the $N$ linear constraints make it even harder to solve exactly because it is known that solving NMF exactly is NP-hard (Vavasis 2009). Therefore, instead of looking for a global optimal solution, we seek to find a local optimal solution that is tractable to compute and at the same time close to a global optimum.

4) The optimal solution $W^*$ is not unique. Given an optimal solution $W^*$, we can construct another optimal solution $\hat{W}^* = W^*\Pi$, where $\Pi$ is a $K \times K$ permutation matrix.[3] The

[3] A permutation matrix is a square binary matrix that has exactly one entry 1 in each row and each column and 0s elsewhere.

effect of right multiplication of $\Pi$ is to permute the labels of the clusters. Considering all possible permutations of cluster labels, there are at least $K!$ optimal solutions. However, any one of the optimal solutions suffices since we do not care about the exact labeling of each cluster.

5) In clustering, we often prefer a sparse solution matrix $W^*$ since only the data points near the boundary of multiple clusters may have a dense probability vector, which means that the majority of the row vectors in $W^*$ should be sparse. This observation will help us to construct an equivalent characterization of (3) in the next section. Furthermore, we can choose a sparse initial matrix $W^{(0)}$ as a starting point for our iterative algorithm introduced in the next section to optimize the objective function based on the sparsity nature of $W$.

## Optimization Algorithm

In this section, we focus on solving the problem in (3) formulated as a constrained norm-minimization problem. As discussed in the previous section, it is difficult to solve this problem exactly. Hence, we seek an approximation procedure by converting the original constrained optimization problem into a sequence of unconstrained optimization problems by using a penalty method. More concretely, we transform the two constraints in (3) into penalty functions that are added to the objective to penalize solutions that are not feasible:

$$\begin{aligned}\text{minimize}_W \quad & ||P - WW^T||_F^2 - \lambda_1 \textstyle\sum_{ij} \min\{0, W_{ij}\} \\ & +\lambda_2 ||W\mathbf{1}_K - \mathbf{1}_N||_2^2\end{aligned} \tag{4}$$

The second term in (4) is a hinge-loss that penalizes elements in $W$ which do not lie in the nonnegative cone. One can also use the log-barrier function $-\lambda_1 \sum_{ij} \log(W_{ij})$ as the penalty term. Here we choose the hinge-loss rather than the log-barrier because we seek a sparse solution (the log-barrier function often pushes the solution away from the boundary of the feasible region until the coefficient $\lambda_1 \to 0$). The third term in (4) is an $\ell_2$ regularization term to penalize rows of $W$ that violate the linear constraints.

In (4), $\lambda_1, \lambda_2 \in \mathbb{R}_+$ determine the tradeoff between the accuracy of the approximation procedure and the feasibility of the solution. With $\lambda_1$ and $\lambda_2 \to \infty$, any violations of the constraints will be greatly penalized and the solution is guaranteed to be in the feasible region. On the other hand, the objective term $||P - WW^T||_F^2$ may be dominated by these two penalty terms, causing the solution to be inaccurate.

To combine the best of two worlds, we propose a sequential minimization framework: for each fixed pair $(\lambda_1, \lambda_2)$, we optimize (4) to find a local minimum solution $W(\lambda_1, \lambda_2)$ which depends on the current $(\lambda_1, \lambda_2)$, then we increase the parameters $(\lambda_1, \lambda_2)$ by a factor $\mu > 1$ and restart the optimization with $W(\lambda_1, \lambda_2)$ as the initial point. To increase the approximation accuracy when $\lambda_1$ and $\lambda_2$ get larger, we increase the number of steps per iteration, ensuring that the penalty terms will vanish as the solution becomes feasible, and the objective term will get fully optimized. The whole process is repeated until the current solution is a stationary point.

To apply the sequential minimization framework, we begin by deriving the gradient for the unconstrained objective function. Throughout this paper we will use the numerator layout (Minka 2000) to compute the differentials, then the gradient is simply the transpose of the corresponding differential. First consider the differential of $||P - WW^T||_F^2$ with respect to $W$:

$$\begin{aligned} d||P - WW^T||_F^2 &= d\text{tr}\{(P^T - WW^T)(P - WW^T)\} \\ &= 2\text{tr}\{(WW^T - P^T)dWW^T\} \\ &= 4\text{tr}\{W^T(WW^T - P^T)dW\} \end{aligned}$$

Note that we are using the fact $\text{tr}(AB) = \text{tr}(BA)$ together with $\text{tr}(A^T) = \text{tr}(A)$. Hence the gradient of the first term in (4) is:

$$\nabla_W\{||P - WW^T||_F^2\} = 4(WW^T - P)W \tag{5}$$

Using the same trick we can compute the gradients of the second term and third term in (4) as:

$$\nabla_W\{-\lambda_1 \sum_{ij} \min\{0, W_{ij}\}\} = -\lambda_1(\mathbb{I}_{W \prec 0} \odot \mathbf{1}_N \mathbf{1}_K^T) \tag{6}$$

and

$$\nabla_W\{\lambda_2 ||W\mathbf{1}_K - \mathbf{1}_N||_2^2\} = 2\lambda_2(W\mathbf{1}_K\mathbf{1}_K^T - \mathbf{1}_N\mathbf{1}_K^T) \tag{7}$$

where $\odot$ is the Hadamard product of two matrices, i.e., the element-wise product, and $\mathbb{I}_{W \prec 0}$ is the indicator function $\mathbb{I}_{x<0}$ applies to every element of $W$.

The gradient of the unconstrained optimization objective is the combination of the three gradients above. Once we obtain a closed form solution to compute the gradient matrix, we can apply various kinds of unconstrained minimization algorithms, e.g., gradient descent, conjugate gradient or L-BFGS (Andrew and Gao 2007) to obtain a local minimum of (4). For illustrative purposes, we provide an algorithm using gradient descent in Alg. 1. Assuming the algorithm will perform $\tau$ iterations in each inner loop, the time complexity of Alg. 1 is $\mathcal{O}(\tau N^2 K \log_\mu(\lambda/\epsilon))$.

## Experiments

We first review how to construct a reasonable distance function $L$ required in Alg. 1. To demonstrate our algorithm, we first apply SoF on synthetic data, which consists of both convex and non-convex patterns, and then we compare SoF with multiple clustering algorithms on real-world datasets.

### Distance Function

Our probabilistic clustering algorithm requires a distance function $L(\cdot, \cdot)$ over all pairs of points as an input. As mentioned before, the choice of scale parameter $c$ will have an impact on the final clustering and furthermore is problem dependent. To avoid an annoying tuning process, we propose to use a *relative distance* $L'(\cdot, \cdot)$ induced from the *absolute distance* $L(\cdot, \cdot)$. More specifically, for any two points $\mathbf{v}_i, \mathbf{v}_j \in I$, we define $L'(\mathbf{v}_i, \mathbf{v}_j)$ as $L'(\mathbf{v}_i, \mathbf{v}_j) = \frac{L(\mathbf{v}_i, \mathbf{v}_j)}{\sqrt{\sigma_{in}}\sqrt{\sigma_{jn}}}$, where $\sigma_{in}$ is the distance between $\mathbf{v}_i$ and the $n$th closest point of $\mathbf{v}_i$ measured by $L$. Intuitively, $\sigma_{in}$ defines the scope

Table 1: Connections and differences among clustering algorithms

| | Kernel Kmeans | Spectral Clustering | SymNMF | SoF |
|---|---|---|---|---|
| Objective | $\min \|\|K - WW^T\|\|_F^2$ | $\min \|\|L - WW^T\|\|_F^2$ | $\min \|\|A - WW^T\|\|_F^2$ | $\min \|\|P - WW^T\|\|_F^2$ |
| Property | $K$ is S.P.D. | $L$ is graph Laplacian, S.P.D. | $A$ is similarity matrix | $P$ is nonnegative, S.P.D. |
| Constraint | $W^TW = I, W \succeq 0$ | $W^TW = I$ | $W \succeq 0$ | $W \succeq 0, W\mathbf{1} = \mathbf{1}$ |

**Algorithm 1** Sequential Minimization to Unconstrained Problem

**Input:** $I = \{v_1, \ldots, v_N\}$, distance function $L$, # of clusters $K$, scale factor $c > 0$, initial penalty parameters $\lambda_1^{(0)}$, $\lambda_2^{(0)} > 0$, step-factor $\mu > 1$, initial learning rate $0 < \gamma^{(0)} < 1$, threshold $0 < \epsilon_1, \epsilon_2 \ll 1$.

**Output:** $K$ dimensional probability vector $p_v \; \forall v \in I$.

Build distance matrix $D$ such that $D_{ij} = L(v_i, v_j)$.
Build soft-cluster matrix $P$ such that $P_{ij} = e^{-cD_{ij}}$.
Initialize matrix $W^{(0)} \in \mathbb{R}_+^{N \times K}$.
**repeat**
  $W_0^{(t)} \leftarrow W_*^{(t-1)}$
  **repeat**
    Compute the gradient $\nabla_W f(W_l^{(t)})$ according to (5), (6) and (7).
    $W_{l+1}^{(t)} \leftarrow W_l^{(t)} - \gamma^{(t)} \nabla_W f(W_l^{(t)})$.
  **until** $||W_{l+1}^{(t)} - W_l^{(t)}|| < \epsilon_1$.
  $W_*^{(t)} \leftarrow W_L^{(t)}$
  $\lambda_1^{(t+1)} \leftarrow \lambda_1^{(t)} \times \mu$.
  $\lambda_2^{(t+1)} \leftarrow \lambda_2^{(t)} \times \mu$.
  $\gamma^{(t+1)} \leftarrow \gamma^{(t)} / \mu$.
**until** $\lambda_1^{(t)} > 1/\epsilon_2$ and $\lambda_2^{(t)} > 1/\epsilon_2$
**return** $W^*$

of $\mathbf{v}_i$, which, to some extent, converts the absolute distance between two points into a relative distance. This transformation will also give us the *scaling invariant* property in the sense that enlarging or shrinking distances between pairs of points by a constant factor will not affect the clustering result. More specifically, consider the distance $L(\cdot, \cdot)$ between a pair of points that have been scaled by a constant factor $\alpha > 0$, the relative distance will still remain the same:

$$L'(\mathbf{v}_i, \mathbf{v}_j) = \frac{\alpha L(\mathbf{v}_i, \mathbf{v}_j)}{\sqrt{\alpha\sigma_{in}}\sqrt{\alpha\sigma_{jn}}} = \frac{L(\mathbf{v}_i, \mathbf{v}_j)}{\sqrt{\sigma_{in}}\sqrt{\sigma_{jn}}} = L'(\mathbf{v}_i, \mathbf{v}_j)$$

In our experiments, we choose $L(\cdot, \cdot)$ to be the Euclidean distance between each pair of points and set $n = 10$. Different values of $n$ will still affect the clustering, but it will not affect it too much in practice.

## Synthetic Dataset

To visualize the performance of SoF, we perform experiments on the synthetic data set used in (Zelnik-Manor and Perona 2004) and generated by (Pei and Zaïane 2006)[4]. Due to the probabilistic nature of SoF, for each point $\mathbf{v}$, we pick its label by maximizing the conditional probability $p(c = k|\mathbf{v}), c = 1, \ldots, K$. We break ties randomly.

**Clustering Quality** The 6 patterns shown in Fig. 1 have different cluster structures, and some of them are not convex, e.g., 1, 2, 3 and 6. The clustering results are shown in Fig. 1, where different colors correspond to different clusters found by SoF. Since SoF does not make any density assumption, it is able to detect various cluster structures, including convex and non-convex patterns. The fifth pattern in Fig. 1 is generated from 2 Gaussians. However, with the number of clusters $K$ set to be 3, our algorithm is still able to find a reasonable clustering and to quantify the uncertainties along the cluster gap (as shown in Fig. 2).

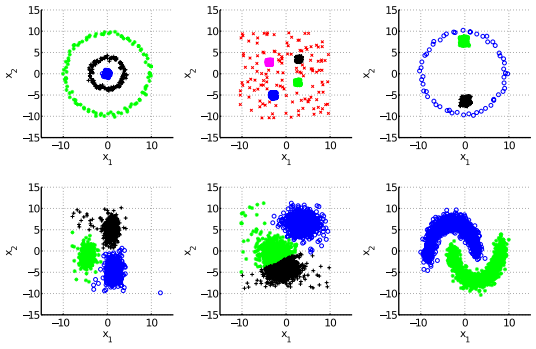

Figure 1: 6 synthetic patterns, where the # of clusters $K$ from left to right and top to bottom is (1) $K = 3$, (2) $K = 5$, (3) $K = 3$, (4) $K = 3$, (5) $K = 3$, (6) $K = 2$.

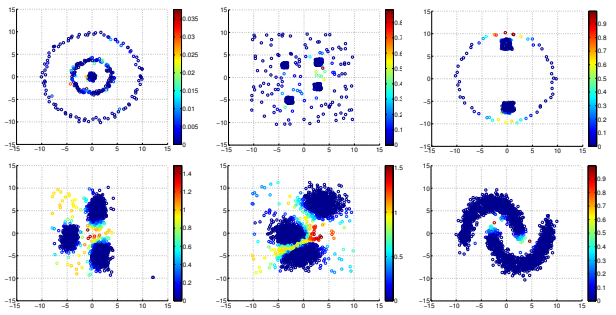

Figure 2: Entropy graphs for 6 patterns in Fig. 1.

[4] http://webdocs.cs.ualberta.ca/~yaling/Cluster/Php/data_gen.php

**Clustering Boundary by Quantifying Uncertainties** Given a probability vector $\mathbf{p} \in \mathbb{S}^{K-1}$, we use *entropy* to measure the uncertainty of cluster assignments of the corresponding data points: $\textit{entropy}(\mathbf{p}) = -\sum_{i=1}^{K} p_i \log p_i$. Hence, points on the cluster boundaries will have a higher entropy while those that are inside a cluster should have lower entropy. Fig. 2 shows the entropy graphs where brighter colors, including yellow and red, indicate a higher entropy value while dark colors such as blue and dark green indicate a lower entropy value. We can visualize the cluster boundaries by checking points associated with brighter colors in convex patterns, e.g. 4 and 5. Furthermore, it can be seen that most of the points (with dark blue colors) belong to a single cluster, indicating that the solutions obtained by SoF are sparse, which is a desirable property in clustering.

## Real-world Dataset

Table 2: Statistics of Datasets

| Data sets | # Objects | # Attributes | # Classes |
|---|---|---|---|
| *blood (BL)* | 748 | 4 | 2 |
| *breast_t (BT)* | 106 | 9 | 6 |
| *glass (GL)* | 214 | 9 | 6 |
| *iris (IRIS)* | 150 | 4 | 3 |
| *ecoli (ECO)* | 336 | 7 | 8 |
| *satimage (IMG)* | 4435 | 36 | 6 |
| *pendigits (DIG)* | 10992 | 16 | 10 |

**Experimental Setup** We use 7 real-world data sets from the UCI Machine Learning Repository[5]. The statistics of the datasets are summarized in Table 2. More detailed information about these data sets can be found at the UCI Machine Learning Repository. The points in each dataset are associated with true labels, which will be used as ground truth for the evaluation of different clustering algorithms.

To evaluate the performance of SoF and compare it with other hard clustering algorithms, we assign each point to the cluster with the highest probability: $l_{\mathbf{v}} = \arg\max_c p(c|\mathbf{v})$. It is commonly admitted that the validation of clustering structures is the most difficult task of clustering analysis (Jain and Dubes 1988). In our experiments we use three robust measures to evaluate clustering performance, namely *purity*, *rand index* and *accuracy* (Jain and Dubes 1988; Cai, He, and Han 2011; Murphy 2012).

1. *Purity*. Let $N_{ij}$ be the number of objects in cluster $i$ that belong to class $j$, and let $N_i = \sum_{j=1}^{C} N_{ij}$ be the total number of objects in cluster $i$. Define $p_{ij} = N_{ij}/N_i$; this is the empirical probability that a point in cluster $i$ is from class $j$. The *purity* of a cluster is defined as $p_i \triangleq \max_j p_{ij}$, and the overall purity of a clustering is defined as
$$\textit{purity} \triangleq \sum_i \frac{N_i}{N} p_i$$
The purity ranges from 0 to 1, and the higher the purity, the better the clustering result.

2. *Rand Index*. Let $U = \{u_1, \ldots, u_R\}$ and $V = \{v_1, \ldots, v_C\}$ be two different partitions of the $N$ data points, i.e., two different flat clusterings. We assume $V$ to be the ground truth clustering, which is induced by class labels. Now define a $2 \times 2$ contingency table, containing the following numbers: TP, TN, FP and FN. TP is the number of pairs that are in the same cluster in both $U$ and $V$ (true positives); TN is the number of pairs that are in the different clusters in both $U$ and $V$ (true negatives); FN is the number of pairs that are in the different clusters in $U$ but the same cluster in $V$ (false negative); and FP is the number of pairs that are in the same cluster in $U$ but different clusters in $V$ (false positive). The *Rand Index* is defined as:
$$R \triangleq \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}}$$
Rand index can be interpreted as the fraction of correctly clustered pairs. Clearly $0 \leq R \leq 1$, and the larger the rand index, the better the clustering.

3. *Accuracy*. Let $N$ be the total number of objects. For the $i$th object, we use $u_i$ to denote the label generated by the clustering algorithm and $v_i$ to denote the ground truth class label associated with the $i$th object. The *accuracy* is defined as
$$\textit{accuracy} = \frac{\delta(v_i, map(u_i))}{N}$$
where $\delta(\cdot,\cdot)$ is the Kronecker delta function, $map(\cdot)$ is the permutation that maps each cluster label generated by the clustering algorithm to a class label of the data corpus. The best mapping can be obtained by Kuhn-Munres' algorithm (Lovász and Plummer 1986). The higher the accuracy, the better the clustering result.

We compare the clustering qualities based on purity, rand index and accuracy of the following clustering algorithms:

1. *Standard K-means* (K-means) and *Spherical K-means* (S-means). For two data points $\mathbf{v}_i$ and $\mathbf{v}_j$, standard K-means uses the Euclidean distance $||\mathbf{v}_i - \mathbf{v}_j||_2$ as its distance function while spherical K-means uses $1 - \cos(\mathbf{v}_i, \mathbf{v}_j)$ as its distance function.
2. *Gaussian Mixture Model* (GMM). EM algorithm is used to fit data density distributions and the number of components is given by the true number of clusters in the data set.
3. *Spectral clustering-Normalized Cut* (SP-NC) and *Spectral clustering-Ng* (SP-NJW). There are two streams of spectral clustering algorithms, namely normalized cut, proposed by Shi and Malik, and Ng-Jordan algorithm, proposed by Ng et al.. In both algorithms, K-means is used as the final procedure to obtain the low-dimensional clustering.
4. *Standard NMF* and *Regularized NMF* (RNMF). Nonnegative matrix factorization based methods, including NMF and its regularized variant, work on a sample matrix directly to obtain a low dimensional representation of the original high dimensional input data. These algorithms interpret each data point by its low dimensional approximation and assign each data point its cluster label by

[5] archive.ics.uci.edu/ml/datasets.html

maximal approximation coefficient in the lower dimensional space. In our experiments, we use the alternating non-negative least square algorithm (Kim and Park 2008a; 2008c). Based on standard NMF, regularized NMF adds two Frobenius regularized terms to penalize non-negative matrix factorization with large elements in the two resulting matrices.

5. *GNMF*. Cai et al. proposed GNMF to take neighboring relationships into account by adding a graph-theoretic penalty term to standard NMF.
6. *SymNMF*. Kuang, Park, and Ding proposed SymNMF as a Newton-like algorithm to solve symmetric non-negative matrix factorization problems, including graph clustering problems. Different from our probabilistic clustering algorithm, SymNMF factorizes the *Graph Laplacian Matrix* (Shi and Malik 2000; Ng et al. 2002) and interprets the result as NMF does.
7. *SoF*. Our proposed probabilistic clustering algorithm using sequential gradient descent minimization.

Table 3: Average purity over 20 runs on 7 UCI data sets.

| | *BL* | *BT* | *GL* | *IRIS* | *ECO* | *IMG* | *DIG* |
|---|---|---|---|---|---|---|---|
| K-means | 0.76 | 0.43 | 0.56 | 0.87 | 0.80 | 0.70 | 0.71 |
| S-means | 0.76 | 0.43 | 0.57 | 0.92 | 0.76 | 0.63 | 0.71 |
| GMM | 0.77 | 0.46 | 0.52 | 0.89 | 0.81 | 0.74 | 0.70 |
| SP-NC | **0.78**$^\dagger$ | 0.42 | 0.38 | 0.86 | 0.81 | 0.30 | 0.11 |
| SP-NJW | 0.76 | 0.40 | 0.38 | 0.59 | 0.80 | 0.48 | 0.13 |
| NMF | 0.76 | 0.43 | 0.58 | 0.79 | 0.73 | 0.57 | 0.47 |
| RNMF | 0.76 | 0.28 | 0.52 | 0.84 | 0.70 | 0.56 | 0.45 |
| GNMF | 0.76 | 0.44 | 0.55 | 0.85 | 0.77 | 0.71 | 0.71 |
| SymNMF | 0.76 | 0.47 | 0.60 | 0.88 | 0.81 | **0.77**$^\dagger$ | 0.76 |
| SoF | 0.76 | **0.51**$^\dagger$ | **0.64**$^\dagger$ | **0.95** | **0.85**$^\dagger$ | 0.75 | **0.82**$^\dagger$ |

Table 4: Average rand index over 20 runs on 7 UCI data sets.

| | *BL* | *BT* | *GL* | *IRIS* | *ECO* | *IMG* | *DIG* |
|---|---|---|---|---|---|---|---|
| K-means | 0.60 | 0.71 | 0.69 | 0.86 | 0.81 | 0.81 | 0.91 |
| S-means | 0.52 | 0.76 | 0.69 | 0.92 | 0.78 | 0.81 | 0.91 |
| GMM | 0.57 | 0.76 | 0.64 | 0.88 | **0.85** | 0.83 | 0.91 |
| SP-NC | **0.64** | 0.74 | 0.43 | 0.85 | 0.81 | 0.32 | 0.11 |
| SP-NJW | 0.51 | 0.71 | 0.53 | 0.60 | 0.80 | 0.74 | 0.41 |
| NMF | 0.56 | 0.65 | 0.71 | 0.81 | 0.81 | 0.80 | 0.86 |
| RNMF | 0.59 | 0.38 | 0.65 | 0.85 | 0.78 | 0.79 | 0.86 |
| GNMF | 0.60 | 0.68 | 0.70 | 0.85 | 0.80 | 0.77 | 0.91 |
| SymNMF | 0.62 | 0.77 | 0.70 | 0.88 | 0.83 | 0.83 | 0.91 |
| SoF | **0.64** | **0.79**$^\dagger$ | **0.73**$^\dagger$ | **0.93** | **0.85** | **0.86**$^\dagger$ | **0.94**$^\dagger$ |

**Results and Analysis** Experimental results on 7 UCI data sets are reported in Tables 3, 4 and 5. We run each clustering algorithm 20 times on each data set and report the average performance. The algorithms which achieve the highest mean over 20 runs are highlighted with bold font and those that are statistically better according to the Wilcoxon signed-rank test (Wilcoxon and Wilcox 1964) with $p$-value $\leq 0.05$ are marked with a dagger, †. Among all the algorithms, K-means and S-means are the fastest. As the algorithm is currently implemented, SoF is slower than previous NMF based clustering methods, including NMF, RNMF, GNMF and SymNMF because of SoF's sequential minimization framework. One possible direction to make SoF more scalable is to utilize the potential sparse structure of the empirical pairwise co-cluster matrix.

Table 5: Average accuracy over 20 runs on 7 UCI data sets.

| | *BL* | *BT* | *GL* | *IRIS* | *ECO* | *IMG* | *DIG* |
|---|---|---|---|---|---|---|---|
| K-means | 0.73 | 0.34 | 0.51 | 0.86 | 0.60 | 0.63 | 0.68 |
| S-means | 0.61 | 0.39 | **0.52** | 0.90 | 0.56 | 0.61 | 0.67 |
| GMM | 0.66 | 0.43 | 0.46 | 0.86 | 0.68 | 0.69 | 0.67 |
| SP-NC | 0.73 | 0.41 | 0.37 | 0.85 | 0.61 | 0.28 | 0.11 |
| SP-NJW | 0.56 | 0.42 | 0.33 | 0.56 | 0.58 | 0.37 | 0.12 |
| NMF | 0.68 | 0.39 | 0.50 | 0.79 | 0.66 | 0.53 | 0.44 |
| RNMF | 0.71 | 0.27 | 0.51 | 0.84 | 0.59 | 0.52 | 0.41 |
| GNMF | 0.72 | 0.36 | 0.44 | 0.81 | 0.56 | 0.54 | 0.67 |
| SymNMF | **0.76** | 0.42 | 0.46 | 0.87 | 0.66 | 0.68 | 0.67 |
| SoF | **0.76** | **0.48**$^\dagger$ | 0.47 | **0.94** | **0.74**$^\dagger$ | **0.71**$^\dagger$ | **0.82**$^\dagger$ |

Since the datasets from different domains present different patterns, our probabilistic clustering algorithm is not biased to a specific domain and it achieves superior results on all three measures in nearly all of the clustering tasks. For some tasks, the comparative algorithms may happen to make the right assumption about the underlying data density and thus yield better results (e.g., spectral clustering is the best for *blood*), but these assumptions may hurt the performance when they do not hold in other datasets (e.g., spectral clustering performs badly on *satimage* and *pendigits*). Furthermore, SoF consistently outperforms other NMF based algorithms, including NMF, RNMF and GNMF, on almost all the datasets.

## Conclusions

In this paper we propose a novel probabilistic clustering algorithm whose objective is axiomatically derived from a set of properties characterizing the co-cluster probability in terms of pairwise distances. Our approach can be viewed as a relaxation of C.P. programming, which intrinsically reveals a close relationship between probabilistic clustering and symmetric NMF-based algorithms. We further design a sequential minimization framework to find a local minimum solution. Experiments on synthetic and real-world datasets show encouraging results. Future work includes speeding up our sequential minimization framework by utilizing the potential sparsity structure of the empirical pairwise co-cluster matrix so that it scales linearly with the non-zero elements in the matrix.

## Acknowledgments

The authors are grateful to Da Kuang for helpful discussions as well as Hongfu Liu for their datasets. The work done by Han Zhao and Pascal Poupart was supported by NSERC Discovery and Engage grants. Han Zhao was also supported by David R. Cheriton Scholarship. Yongfeng Zhang was supported by Baidu and IBM PhD Fellowship program.

## References


Aggarwal, C. C., and Reddy, C. K. 2013. *Data Clustering: Algorithms and Applications*. CRC Press.

Andrew, G., and Gao, J. 2007. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, 33–40. ACM.

Berman, A., and Shaked-Monderer, N. 2003. *Completely positive matrices*. World Scientific.

Biernacki, C.; Celeux, G.; and Govaert, G. 2000. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(7):719–725.

Cai, D.; He, X.; Han, J.; and Huang, T. S. 2011. Graph regularized nonnegative matrix factorization for data representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(8):1548–1560.

Cai, D.; He, X.; and Han, J. 2011. Locally consistent concept factorization for document clustering. *Knowledge and Data Engineering, IEEE Transactions on* 23(6):902–913.

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2004. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 551–556. ACM.

Dickinson, P. J., and Gijben, L. 2014. On the computational complexity of membership problems for the completely positive cone and its dual. *Computational Optimization and Applications* 57(2):403–415.

Ding, C. H.; He, X.; and Simon, H. D. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*, volume 5, 606–610.

Ding, C.; Li, T.; and Peng, W. 2008. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis* 52(8):3913–3927.

Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pretraining help deep learning? *The Journal of Machine Learning Research* 11(3):625–660.

He, Z.; Xie, S.; Zdunek, R.; Zhou, G.; and Cichocki, A. 2011. Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering. *Neural Networks, IEEE Transactions on* 22(12):2117–2131.

Jain, A. K., and Dubes, R. C. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

Jain, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters* 31(8):651–666.

Kim, H., and Park, H. 2008a. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications* 30(2):713–730.

Kim, J., and Park, H. 2008b. Sparse nonnegative matrix factorization for clustering.

Kim, J., and Park, H. 2008c. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 353–362. IEEE.

Kleinberg, J. 2003. An impossibility theorem for clustering. *Advances in neural information processing systems* 463–470.

Kuang, D.; Park, H.; and Ding, C. H. 2012. Symmetric nonnegative matrix factorization for graph clustering. In *SDM*, volume 12, 106–117.

Lovász, L., and Plummer, M. D. 1986. Matching theory. *New York*.

Minka, T. P. 2000. Old and new matrix algebra useful for statistics. *See www. stat. cmu. edu/minka/papers/matrix. html*.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. The MIT Press.

Ng, A. Y.; Jordan, M. I.; Weiss, Y.; et al. 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 2:849–856.

Pei, Y., and Zaïane, O. 2006. A synthetic data generator for clustering and outlier analysis. *Department of Computing science, University of Alberta, edmonton, AB, Canada*.

Seung, D., and Lee, L. 2001. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* 13:556–562.

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

Steinbach, M.; Karypis, G.; Kumar, V.; et al. 2000. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, 525–526. Boston.

Thalamuthu, A.; Mukhopadhyay, I.; Zheng, X.; and Tseng, G. C. 2006. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics* 22(19):2405–2412.

Vavasis, S. A. 2009. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization* 20(3):1364–1377.

Vogt, J. E.; Prabhakaran, S.; Fuchs, T. J.; and Roth, V. 2010. The translation-invariant wishart-dirichlet process for clustering distance data. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1111–1118.

Wilcoxon, F., and Wilcox, R. A. 1964. *Some rapid approximate statistical procedures*. Lederle Laboratories.

Xu, W.; Liu, X.; and Gong, Y. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 267–273. ACM.

Zelnik-Manor, L., and Perona, P. 2004. Self-tuning spectral clustering. In *Advances in neural information processing systems*, 1601–1608.