# Laplacian Regularized Gaussian Mixture Model for Data Clustering

Xiaofei He, *Member*, *IEEE*, Deng Cai, Yuanlong Shao, Hujun Bao, and Jiawei Han, *Fellow*, *IEEE*

**Abstract**—Gaussian Mixture Models (GMMs) are among the most statistically mature methods for clustering. Each cluster is represented by a Gaussian distribution. The clustering process thereby turns to estimate the parameters of the Gaussian mixture, usually by the Expectation-Maximization algorithm. In this paper, we consider the case where the probability distribution that generates the data is supported on a submanifold of the ambient space. It is natural to assume that if two points are close in the intrinsic geometry of the probability distribution, then their conditional probability distributions are similar. Specifically, we introduce a regularized probabilistic model based on manifold structure for data clustering, called *Laplacian regularized Gaussian Mixture Model* (LapGMM). The data manifold is modeled by a nearest neighbor graph, and the graph structure is incorporated in the maximum likelihood objective function. As a result, the obtained conditional probability distribution varies smoothly along the geodesics of the data manifold. Experimental results on real data sets demonstrate the effectiveness of the proposed approach.

**Index Terms**—Gaussian mixture model, clustering, graph laplacian, manifold structure.

## 1 Introduction

The goal of data clustering is to group a collection of objects into subsets such that those within each cluster are more closely related to one another than objects assigned to different clusters. Data clustering is a common technique for exploratory data analysis, which is used in many fields, including data mining, machine learning, pattern recognition, and information retrieval. It has received enormous attention in recent years [2], [4], [9], [10], [15], [19], [20], [28], [29], [46].

The clustering algorithms can be roughly categorized into similarity-based and model-based. Similarity-based clustering algorithms require no assumption on probability structure of the data. It only requires a similarity function defined on the data pairs. The typical similarity-based algorithms include $K$-means [24] and spectral clustering [39], [42], [10], [47]. $K$-means produces a cluster set that minimizes the sum of squared errors between the data points and the cluster centers. The spectral clustering usually clusters the data points using the top eigenvectors of *graph Laplacian* [18], which is defined on the affinity matrix of data points. From the graph partitioning perspective, spectral clustering tries to find the best cut of the graph so that the predefined criterion function can be optimized. Many criterion functions, such as ratio cut [16], average association [42], normalized cut [42], and min-max cut have been proposed along with the corresponding eigen-problem for finding their optimal solutions. Spectral clustering has achieved great success in many real world applications, such as image segmentation [42], Web image clustering [11], community mining [1], and bioinformatics [25].

Besides spectral clustering, matrix factorization-based approaches for data clustering have been proposed recently. Out of them, Nonnegative Matrix Factorization (NMF) is the most popular one [30], [47], [31]. NMF aims to find two nonnegative matrices whose product provides a good approximation to the original matrix. The nonnegative constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. This way, NMF models each cluster as a linear combination of the data points, and each data point as a linear combination of the clusters.

Unlike similarity-based clustering which generates hard partition of data, model-based clustering can generate soft partition which is sometimes more flexible. Model-based methods use mixture distributions to fit the data and the conditional probabilities of data points are naturally used to assign probabilistic labels. One of the most widely used mixture models for clustering is Gaussian Mixture Model [7]. Each Gaussian density is called a component of the mixture and has its own mean and covariance. In many applications, their parameters are determined by maximum likelihood, typically using the Expectation-Maximization algorithm [21].

Recently, various researchers (see [43], [40], [5], [41], [6]) have considered the case when the data is drawn from sampling a probability distribution that has support on or near to a *submanifold* of the ambient space. Here, a $d$-dimensional submanifold of a euclidean space $\mathbb{R}^n$ is a subset $\mathcal{M}^d \subset \mathbb{R}^n$ which locally looks like a flat $d$-dimensional euclidean space [32]. For example, a sphere is a two-dimensional submanifold of $\mathbb{R}^3$. In order to detect the underlying manifold structure, many *manifold learning*

- *X. He, D. Cai, Y. Shao, and H. Bao are with the State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, 388 Yu Hang Tang Rd., Hangzhou, Zhejiang 310058, China. E-mail: {xiaofeihe, dengcai, shaoyuanlong, bao}@cad.zju.edu.cn.*
- *J. Han is with the Department of Computer Science, University of Illinois at Urbana Champaign, Siebel Center, 201 N. Goodwin Ave., Urbana, IL 61801. E-mail: hanj@cs.uiuc.edu.*

*Manuscript received 11 Aug. 2008; revised 10 Feb. 2009; accepted 30 Oct. 2009; published online 22 Dec. 2010.*
*Recommended for acceptance by D. Talia.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-08-0418.*
*Digital Object Identifier no. 10.1109/TKDE.2010.259.*

algorithms have been proposed, such as Locally Linear Embedding [40], [35], Isomap [43], and Laplacian eigenmap [5]. The basic idea of LLE is that the data points might reside on a nonlinear submanifold, but it might be reasonable to assume that each local neighborhood is linear. Thus, we can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. Laplacian eigenmap is based on spectral graph theory [18]. It finds a nonlinear mapping by discretely approximating the eigenfunctions of the Laplace-Beltrami operator on the manifold. This way, the obtained mapping preserves the local structure. Isomap aims at finding a euclidean embedding such that euclidean distances in $\mathbb{R}^n$ can provide a good approximation to the geodesic distances on the manifold. A *geodesic* is a generalization of the notion of a "straight line" to "curved spaces." It is defined to be the shortest path between points on the space. On the sphere, the geodesics are great circles (like the equator). Isomap is a global method which attempts to preserve geometry at all scales, mapping nearby points on the manifold to nearby points in low-dimensional space, and faraway points to faraway points. Most of the current manifold learning techniques focus on dimensionality reduction [43], [40], [5], [48], and semisupervised learning [6]. It has been shown that learning performance can be significantly enhanced if the geometrical structure is exploited.

In this paper, we propose a novel model-based algorithm for data clustering, called *Laplacian regularized Gaussian Mixture Model* (LapGMM), which explicitly considers the manifold structure. Following the intuition that naturally occurring data may reside on or close to a *submanifold* of the ambient space, we aim to fit a probabilistic model while respecting the manifold structure. Specifically, if two data points are sufficiently close on the manifold, then they are likely to be generated by sampling the same Gaussian component. In real world applications, the data manifold is usually unknown. Therefore, we construct a nearest neighbor graph to discretely model the manifold structure. Using *graph Laplacian* [18], the manifold structure can be incorporated in the log-likelihood function of the standard GMM as a regularization term. This way, we smooth the conditional probability density functions along the geodesics on the manifold, where nearby data points have similar conditional probability density functions. Laplacian regularization has also been incorporated into topic models to analyze text documents [12], [13], [37]. It would be important to note that our proposed algorithm is essentially different from these approaches in that LapGMM aims at estimating the Gaussian mixture density function that generates the data.

It is worthwhile to highlight several aspects of the proposed approach here:

1. While the standard GMM fits the data in euclidean space, our algorithm exploits the intrinsic geometry of the probability distribution that generates the data and incorporates it as an additional regularization term. Hence, our algorithm is particularly applicable when the data is sampled from a submanifold which is embedded in high-dimensional ambient space.
2. Our algorithm constructs a nearest neighbor graph to model the manifold structure. The weight matrix of the graph is highly sparse. Therefore, the computation of conditional probabilities is very efficient. By preserving the graph structure, our algorithm can have more discriminating power than the standard GMM algorithm.
3. The proposed framework is a general one that can leverage the power of both the mixture model and the graph Laplacian regularization. Besides graph Laplacian, there are other smoothing operators which can be used to smooth the conditional probability density function, such as iterated Laplacian, heat semigroup, and Hessian [6].

The rest of the paper is organized as follows: Section 2 gives a brief description of the related work. In Section 3, we introduce our proposed Laplacian regularized Gaussian Mixture Model for data clustering. The experimental results are shown in Section 4. Finally, we provide some concluding remarks and suggestions for future work in Section 5.

## 2 Related Work

In this section, we provide a brief description of the related work.

### 2.1 Gaussian Mixture Model

From a model-based perspective, each cluster can be mathematically represented by a parametric distribution. The entire data set is, therefore, modeled by a mixture of these distributions. The most widely used model in practice is the mixture of Gaussians:

$$P(\mathbf{x}|\Theta) = \sum_{i=1}^{K} \alpha_i p_i(\mathbf{x}|\theta_i),$$

where the parameters are $\Theta = (\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K)$ such that $\sum_{i=1}^{K} \alpha_i = 1$ and each $p_i$ is a Gaussian density function parameterized by $\theta_i$. In other words, we assume that we have $K$ component densities mixed together with $K$ mixing coefficients $\alpha_i$.

Let $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be a set of data points. We wish to find $\Theta$ such that $p(\mathcal{X}|\Theta)$ is a maximum. This is known as the Maximum Likelihood(ML) estimate for $\Theta$. In order to estimate $\Theta$, it is typical to introduce the log likelihood function defined as follows:

$$\begin{aligned}\mathcal{L}(\Theta) = \log P(\mathcal{X}|\Theta) &= \log \prod_{i=1}^{m} P(\mathbf{x}_i|\Theta) \\ &= \sum_{i=1}^{m} \log\left(\sum_{j=1}^{K} \alpha_j p_j(\mathbf{x}_i|\theta_j)\right)\end{aligned}$$

which is difficult to optimize because it contains the log of the sum. To simplify the likelihood expression, let $y_i \in \{1, \ldots, K\}$ denote which Gaussian $\mathbf{x}_i$ is from and $\mathcal{Y} = (y_1, \ldots, y_m)$. If we know the value of $\mathcal{Y}$, the likelihood becomes

$$\begin{aligned}\mathcal{L}(\Theta) = \log P(\mathcal{X}, \mathcal{Y}|\Theta) &= \log \prod_{i=1}^{m} P(\mathbf{x}_i, y_i|\Theta) \\ &= \sum_{i=1}^{m} \log P(\mathbf{x}_i|y_i)P(y_i) = \sum_{i=1}^{m} \log(\alpha_{y_i} p_{y_i}(\mathbf{x}_i|\theta_{y_i}))\end{aligned}$$

which can be optimized using a variety of techniques, such as Expectation-Maximization algorithm.

Comparison of the $K$-means algorithm with the EM algorithm for Gaussian mixtures shows that there is a close similarity [7]. Whereas the $K$-means algorithm performs a *hard* assignment of data points to clusters, in which each data point is associated uniquely with one cluster, the EM algorithm makes a *soft* assignment based on the posterior probabilities. In fact, we can derive the $K$-means algorithm as a particular limit of EM for Gaussian mixtures. For details, please see [7].

### 2.2 Nonnegative Matrix Factorization

NMF is a matrix factorization algorithm that has achieved great success in data clustering [47]. Given a nonnegative data matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$, and each column of $\mathbf{X}$ is a sample vector. NMF aims to find two nonnegative matrices $U = [u_{ik}] \in \mathbb{R}^{m \times t}$ and $V = [v_{jk}] \in \mathbb{R}^{n \times t}$ which minimize the following objective function:

$$O = \|X - UV^T\|^2, \tag{1}$$

where $\|\cdot\|$ denotes the matrix *Frobenius norm*.

In reality, we have $t \ll m$ and $t \ll n$. Thus, NMF essentially try to find a compressed approximation of the original data matrix, $X \approx UV^T$. We can view this approximation column by column as

$$\mathbf{x}_j \approx \sum_{k=1}^{t} \mathbf{u}_k v_{jk}, \tag{2}$$

where $\mathbf{u}_k$ is the $k$th column vector of $U$. Thus, each data vector $\mathbf{x}_j$ is approximated by a linear combination of the columns of $U$, weighted by the components of $V$. Therefore, $U$ can be regarded as containing a basis that is optimized for the linear approximation of the data in $X$. Let $\mathbf{z}_j^T$ denote the $j$th row of $V$, $\mathbf{z}_j = [v_{j1}, \ldots, v_{jk}]^t$. $\mathbf{z}_j$ can be regarded as the new representation of each data point in the new basis $U$. Since relatively few basis vectors are used to represent many data vectors, good approximation can only be achieved if the basis vectors discover structure that is latent in the data [31]. The nonnegative constraints on $U$ and $V$ only allow addictive combinations among different basis. This is the most significant difference between NMF and other matrix factorization methods, e.g., Singular Value Decomposition (SVD). Unlike SVD, no subtractions can occur in NMF. For this reason, it is believed that NMF can learn a *parts-based* representation [30]. The advantages of this parts-based representation has been observed in many real world problems such as face analysis [33], document clustering [47] and DNA gene expression analysis [8]. For more detailed analysis of NMF and its various extensions, please see [17], [22], [23], [26], [34].

## 3 Laplacian Regularized GMM

In this section, we introduce our LapGMM algorithm, which learns a Gaussian mixture model by exploiting the intrinsic geometry of the probability distribution that generates the data points. Since our approach is fundamentally based on differential geometry, we begin with a brief description of the basic geometrical concepts.

### 3.1 Riemannian Manifolds

Manifolds are generalizations of curves and surfaces to arbitrarily many dimensions. In the simplest terms, these are spaces that locally look like some euclidean space $\mathbb{R}^d$, and on which one can do calculus. The most familiar examples, aside from euclidean spaces themselves, are smooth plane curves such as circles and parabolas, and smooth surfaces such as spheres, tori, paraboloids, ellipsoids, and hyperboloids. When the manifold is embedded in a high-dimensional euclidean space $\mathbb{R}^n$, we also call it submanifold of $\mathbb{R}^n$ and this euclidean space is called ambient space. The formal definition of submanifolds is as follows:

**Definition of submanifolds.** *Let $\mathcal{M}$ be a subset of $\mathbb{R}^n$ such that for every point $x \in \mathcal{M}$ there exists a neighborhood $U_x$ of $x$ in $\mathbb{R}^n$ and $d$ continuously differentiable functions $\rho_k : U \to \mathbb{R}$ where the differentials of $\rho_k$ are linearly independent, such that*

$$\mathcal{M} \cap U = \{x \in U \mid \rho_k(x) = 0, 1 \le k \le d\}.$$

*Then $\mathcal{M}$ is called a submanifold of $\mathbb{R}^n$ of dimension $d$.*

In order to calculate the distance between two points on the manifold $\mathcal{M}$, we need to equip it with a Riemannian metric $g$ which is a function that assigns for each point $x \in \mathcal{M}$ a positive definite inner product on the tangent space $T_x\mathcal{M}$. Once the Riemannian metric is defined, one is allowed to measure the lengths of the tangent vectors $\mathbf{v} \in T_x\mathcal{M}$:

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle = \sum_{ij} g_{ij} v^i v^j.$$

Since $g$ is positive definite, $\|\mathbf{v}\|$ is nonnegative. For every smooth curve $r : [a, b] \to \mathcal{M}$, we have tangent vectors

$$r'(t) = \frac{dr}{dt} \in T_{r(t)}\mathcal{M}.$$

We can then define the length of $r$ from $a$ to $b$:

$$\text{length}(r) = \int_a^b \|\frac{dr}{dt}\| dt = \int_a^b \|r'(t)\| dt.$$

A manifold equipped with Riemannian metric is called *Riemannian manifold*. The geodesics are defined as follows:

**Definition of geodesics.** *Geodesics are the shortest path between points on the manifold.*

To distinguish between the submanifold and the ambient euclidean space, we usually use intrinsic geometry to denote the geometrical properties of the submanifold in question.

### 3.2 Gaussian Mixture Model with Manifold Regularization

As we have previously described, naturally occurring data may be generated by structured systems with possibly much fewer degrees of freedom than what the ambient dimension would suggest. Thus, we consider the case when the data lives on or close to a submanifold of the ambient space. In this section, we show how geometric knowledge of the probability distribution may be incorporated in learning a Gaussian mixture model.

Recall the standard framework of learning from examples. There is a probability distribution $P$ on $X \times \mathbb{R}$ according to which examples are generated for function learning. For clustering problem, examples are simply $x \in X$ drawn according to the marginal distribution $\mathcal{P}_X$ of $P$. Previous studies have shown that there may be connection between the marginal and conditional distributions [6]. Specifically, if two points $\mathbf{x}_1, \mathbf{x}_2 \in X$ are close in the intrinsic geometry of $\mathcal{P}_X$, then the conditional probabilities $\mathcal{P}(y|\mathbf{x}_1)$ and $\mathcal{P}(y|\mathbf{x}_2)$ are similar, where $y \in \{1, \ldots, K\}$ is the class label. In other words, the conditional probability distribution $\mathcal{P}(y|\cdot)$ varies smoothly along the geodesics in the intrinsic geometry of $\mathcal{P}_X$. This is usually referred to as *manifold assumption* [6], which plays an essential role in developing various kinds of algorithms including dimensionality reduction [5] and semisupervised learning algorithms [6], [49]. We utilize these geometric intuitions to extend an established framework for learning a Gaussian mixture model.

Suppose there are $K$ components. Let $f_k(\mathbf{x}) = P(y = k|\mathbf{x}) \doteq P(k|\mathbf{x})$ be the conditional Probability Distribution Function, $k = 1, \ldots, K$. We use $\|f_k\|_{\mathcal{M}}$ to measure the smoothness of $f_k$. When we consider the case that the support of $\mathcal{P}_X$ is a compact submanifold $\mathcal{M} \subset \mathbb{R}^d$, a natural choice for $\|f_k\|_{\mathcal{M}}$ is

$$\int_{\mathbf{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f_k\|^2 d\mathcal{P}_X(\mathbf{x}) \tag{3}$$

which is equivalent to

$$\int_{\mathbf{x} \in \mathcal{M}} \mathcal{L}(f_k) f_k d\mathcal{P}_X(\mathbf{x}), \tag{4}$$

where $\nabla_{\mathcal{M}}$ is the gradient of $f_k$ along the manifold and the integral is taken over the distribution $\mathcal{P}_X$. $\mathcal{L}$ is the Laplace-Beltrami operator on the manifold, that is, $\mathcal{L}f = -div\nabla(f)$. By minimizing $\|f_k\|^2_{\mathcal{M}}$, we can obtain a sufficiently smooth conditional probability distribution function.

In reality, however, the data manifold is usually unknown. Thus, $\|f_k\|^2_{\mathcal{M}}$ cannot be computed. In order to model the geometrical structure of $\mathcal{M}$, we construct a nearest neighbor graph $G$. For each data point $\mathbf{x}_i$, we find its $p$ nearest neighbors and put an edge between $\mathbf{x}_i$ and its neighbors. The $p$ nearest neighbor is defined as follows:

**Definition of $p$ nearest neighbors.** *Given $m$ data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^n$. For any point $\mathbf{x}_i$, sort the rest $m-1$ points according to their euclidean distances to $\mathbf{x}_i$, in ascending order. The top-$p$ ranked points are called the $p$ nearest neighbors of $\mathbf{x}_i$.*

There are many choices to define the weight matrix $S$ on the graph. Three of the most commonly used are as follows:

1. **0-1 weighting**. $S_{ij} = 1$ if and only if nodes $i$ and $j$ are connected by an edge. This is the simplest weighting method and is very easy to compute.
2. **Heat kernel weighting**. If nodes $i$ and $j$ are connected, put

$$S_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}.$$

   Heat kernel has an intrinsic connection to the Laplace Beltrami operator on differentiable functions on a manifold [5].

3. **Dot-product weighting**. If nodes $i$ and $j$ are connected, put

$$S_{ij} = \mathbf{x}_i^T \mathbf{x}_j.$$

   Note that, if $\mathbf{x}$ is normalized to 1, the dot product of two vectors is equivalent to the cosine similarity of the two vectors.

Define $L = D - S$, where $D$ is a diagonal matrix whose entries are column (or row, since $S$ is symmetric) sums of $S$, that is, $D_{ii} = \sum_j S_{ij}$. $L$ is called graph Laplacian [18]. By spectral graph theory [18], [5], $\|f_k\|^2_{\mathcal{M}}$ can be discretely approximated as follows:

$$\begin{aligned}
\mathcal{R}_k &= \frac{1}{2}\sum_{i,j=1}^{m} \big(P(k|\mathbf{x}_i) - P(k|\mathbf{x}_j)\big)^2 S_{ij} \\
&= \sum_{i=1}^{m} P(k|\mathbf{x}_i)^2 D_{ii} - \sum_{i,j=1}^{m} P(k|\mathbf{x}_i) P(k|\mathbf{x}_j) S_{ij} \\
&= \mathbf{f}_k^T D \mathbf{f}_k - \mathbf{f}_k^T S \mathbf{f}_k \\
&= \mathbf{f}_k^T L \mathbf{f}_k,
\end{aligned} \tag{5}$$

where

$$\mathbf{f}_k = \big(f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_m)\big)^T = \big(P(k|\mathbf{x}_1), \ldots, P(k|\mathbf{x}_m)\big)^T.$$

By minimizing $\mathcal{R}_k$, we get a conditional distribution $f_k$ which is sufficiently smooth on the data manifold. Specifically, the objective function (5) with our choice of $S_{ij}$ incurs a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ have very different conditional probability distributions. Therefore, minimizing $\mathcal{R}_k$ is an attempt to ensure that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close,"then $f_k(\mathbf{x}_i)$ and $f_k(\mathbf{x}_j)$ are close as well. It is worth emphasizing that the Laplacian of a graph is analogous to the Laplace-Beltrami operator on manifolds.

Now we can define the regularized log-likelihood function as follows:

$$\begin{aligned}
\mathcal{L}(\Theta) &= \log P(\mathcal{X}|\Theta) - \lambda \sum_{k=1}^{K} \mathcal{R}_k \\
&= \sum_{i=1}^{m} \log\left(\sum_{j=1}^{K} \alpha_j p_j(\mathbf{x}_i|\theta_j)\right) - \lambda \sum_{k=1}^{K} \mathcal{R}_k,
\end{aligned} \tag{6}$$

where $\lambda$ is the regularization parameter which controls the smoothness of $f_k (k = 1, \ldots, K)$.

Similar to the standard Gaussian mixture model, the above objective function is very difficult to optimize. Therefore, we introduce latent variables $\mathcal{Y} = (y_1, \ldots, y_m)$. We shall call $\{\mathcal{X}, \mathcal{Y}\}$ the *complete* data set, and we refer to the actual observed data $\mathcal{X}$ as *incomplete*. The Laplacian regularized log-likelihood function for the complete data set takes the form:

$$\begin{aligned}
\mathcal{L}(\Theta) &= \log P(\mathcal{X}, \mathcal{Y}|\Theta) - \lambda \sum_{k=1}^{K} \mathcal{R}_k \\
&= \sum_{i=1}^{m} \log\big(\alpha_{y_i} p_{y_i}(\mathbf{x}_i|\theta_{y_i})\big) - \lambda \sum_{k=1}^{K} \mathcal{R}_k.
\end{aligned} \tag{7}$$

### 3.3 Model Fitting Using Generalized EM

The EM algorithm is used for finding maximum likelihood parameter estimates when there is missing or incomplete

data. In our case, the missing data is the Gaussian cluster to which the data points belong. We estimate values to fill in for the missing data (the E-step), compute the maximum-likelihood parameter estimates using this data (the M-step), and repeat until a suitable stopping criterion is reached. The optimization scheme described here is fundamentally motivated from our previous work [12], [13], [37].

#### 3.3.1 E-Step

The EM algorithm first evaluates the posterior probabilities using the current parameter values $\alpha_i^{n-1}$ and $\theta_i^{n-1}$ $(i = 1, \ldots, K)$:

$$P(k|\mathbf{x}_i, \Theta^{n-1}) = \frac{\alpha_k^{n-1} p_k\left(\mathbf{x}_i|\theta_k^{n-1}\right)}{\sum_{j=1}^{K} \alpha_j^{n-1} p_j\left(\mathbf{x}_i|\theta_j^{n-1}\right)}, \tag{8}$$

where $\alpha_i^{n-1}$ are the mixing coefficients and $\theta_i^{n-1}$ denotes the mean and covariance for the $ith$ component. We then find the expected value of the complete-data log likelihood $\log P(\mathcal{X}, \mathcal{Y}|\Theta)$ with respect to the unknown data $\mathcal{Y}$ given the observed data $\mathcal{X}$ and the current parameter estimates. That is, we define:

$$Q(\Theta, \Theta^{n-1}) = E\left[\log P(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{n-1}\right]. \tag{9}$$

With simple derivations, (9) can be rewritten as follows:

$$\begin{aligned} Q(\Theta, \Theta^{n-1}) = &\sum_{j=1}^{K}\sum_{i=1}^{m} \log(\alpha_j) p(j|\mathbf{x}_i, \Theta^{n-1}) \\ &+ \sum_{j=1}^{K}\sum_{i=1}^{m} \log\left(p_j(\mathbf{x}_i|\theta_j)\right) p(j|\mathbf{x}_i, \Theta^{n-1}). \end{aligned} \tag{10}$$

Incorporating the Laplacian regularization term $\mathcal{R}_i$ into the above expression, thus the regularized expected complete data log-likelihood for LapGMM is as follows:

$$\begin{aligned} &\mathcal{Q}(\Theta, \Theta^{n-1}) \\ &= Q(\Theta, \Theta^{n-1}) - \lambda \sum_{k=1}^{K} \mathcal{R}_k \\ &= \sum_{j=1}^{K}\sum_{i=1}^{m} \log(\alpha_j) p(j|\mathbf{x}_i, \Theta^{n-1}) \\ &\quad + \sum_{j=1}^{K}\sum_{i=1}^{m} \log(p_j(\mathbf{x}_i|\theta_j)) p(j|\mathbf{x}_i, \Theta^{n-1}) \\ &\quad - \lambda \sum_{k=1}^{K}\sum_{i,j=1}^{m} (P(k|\mathbf{x}_i, \Theta) - P(k|\mathbf{x}_j, \Theta))^2 S_{ij}. \end{aligned} \tag{11}$$

#### 3.3.2 M-Step

In M-step, we need to solve the maximization problem (11). However, there is no closed form solution for (11). In the following, we discuss how to apply the generalized EM algorithm (GEM, [38]) to maximize the regularized log-likelihood function of LapGMM. The major difference between GEM and traditional EM is in M-step. Instead of finding the globally optimal solutions for $\Theta$ which maximize (11), if suffices for GEM to find a "better" $\Theta$. Let $\Theta^{n-1}$ denote the parameters of previous iteration and $\Theta^n$ the current parameters. The convergence of the GEM algorithm only requires that $\mathcal{Q}(\Theta^n) \geq \mathcal{Q}(\Theta^{n-1})$. For Gaussian distribution, the parameter $\theta_i = (\mu_i, \Sigma_i)$, where $\mu_i$ is the mean and $\Sigma_i$ is the covariance matrix.

Our basic idea is to maximize $Q(\Theta, \Theta^{n-1})$ and minimize $\sum_{k=1}^{K} \mathcal{R}_k$ separately in hope of finding an improvement of the current $\mathcal{Q}$. Notice that the regularization term only involves posterior probabilities $P(k|\mathbf{x}_i)$. We initialize $P(k|\mathbf{x}_i)$ to the value of previous iteration, that is, $P(k|\mathbf{x}_i, \Theta^{n-1})$, and try to decrease $\sum_{k=1}^{K} \mathcal{R}_k$, which can be done by applying Newton-Raphson method. As stated in Section 3.1, $\mathcal{R}_k$ can be rewritten as $\mathbf{f}_k^T L \mathbf{f}_k$, where $\mathbf{f}_k = (P(k|\mathbf{x}_1), \ldots, P(k|\mathbf{x}_m))^T$ and $L$ is a constant matrix depending on the data.

Given a function $f(x)$ and the initial value $x_t$, the Newton-Raphson updating formula to decrease (or increase) $f(x)$ is as follows:

$$x_{t+1} = x_t - \gamma \frac{f'(x)}{f''(x)},$$

where $0 \leq \gamma \leq 1$ is the step parameter. By taking the first and second derivatives of $\mathbf{f}_k^T L \mathbf{f}$ with respect to $P(k|\mathbf{x}_i)$ (the $i$-$th$ element of $\mathbf{f}_k$), we get the following updating scheme for $P(k|\mathbf{x}_i)$:

$$P(k|\mathbf{x}_i) \leftarrow (1-\gamma) P(k|\mathbf{x}_i) + \gamma \frac{\sum_{j=1}^{m} S_{ij} P(k|\mathbf{x}_j)}{\sum_{j=1}^{m} S_{ij}}, i = 1, \ldots, m. \tag{12}$$

It is easy to show that the graph Laplacian $L$ is positive semidefinite, so each updating step will decrease $\mathcal{R}_k$. Moreover, the weight matrix $S$ is highly sparse, so the updating of $P(k|\mathbf{x}_i)$ is very efficient.

After the conditional probability function is smoothed, we can maximize the log-likelihood function $Q(\Theta, \Theta^{n-1})$ of the standard GMM and get the following updating scheme for Gaussian parameters:

$$\alpha_i = \frac{1}{m} \sum_{j=1}^{m} p(i|\mathbf{x}_j), \tag{13}$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{m} \mathbf{x}_j p(i|\mathbf{x}_j)}{\sum_{j=1}^{m} p(i|\mathbf{x}_j)}, \tag{14}$$

$$\Sigma_i = \frac{\sum_{j=1}^{m} p(i|\mathbf{x}_j)(\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{m} p(i|\mathbf{x}_j)}. \tag{15}$$

In general, the GEM algorithm takes many more iterations to reach convergence compared with the $K$-means algorithm, and each cycle requires significantly more computation. It is, therefore, common to run the $K$-means algorithm in order to find a suitable initiation for our algorithm that is subsequently adapted using GEM. The use of $K$-means for initialization is mostly due to its efficiency. Comparing to random initialization, $K$-means can obviously obtain better result and therefore speedup the convergence. We may also consider other clustering algorithms such as spectral clustering and NMF for initialization. However, these algorithms are themselves computationally expensive and thus not appropriate for

TABLE 1
The LapGMM Algorithm

**The LapGMM Algorithm**

**Input:** $m$ data vectors $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \subset \mathbb{R}^d$, the number of clusters $K$, the number of nearest neighbors $p$, regularization parameter $\lambda$, the termination condition value $\delta$.

**Output:** $\alpha_1, \cdots, \alpha_K$; $\{\boldsymbol{\mu}_1, \Sigma_1\}, \cdots, \{\boldsymbol{\mu}_K, \Sigma_K\}$.

0: Set the initial value $\gamma = 0.9$.
1: Construct a nearest neighbor graph with weight matrix $S$.
2: Initialize the parameters $\Theta^0$ by using $K$-means algorithm.
3: $n \leftarrow 1$;
4: **while** (true)
**E-step**:
5: Compute the posterior probabilities:

$$P(k|\mathbf{x}_i) = \frac{\alpha_k^{n-1} p_k(\mathbf{x}_i|\theta_k^{n-1})}{\sum_{j=1}^{K} \alpha_j^{n-1} p_j(\mathbf{x}_i|\theta_j^{n-1})}$$

**M-step**:
6: Smooth the posterior probabilities until convergence:

$$P(k|\mathbf{x}_i) \Leftarrow (1-\gamma)P(k|\mathbf{x}_i) + \gamma \frac{\sum_{j=1}^{m} S_{ij} P(k|\mathbf{x}_j)}{\sum_{j=1}^{m} S_{ij}}, (i = 1, \cdots, m; k = 1, \cdots, K.)$$

7: Compute the LapGMM estimates $\alpha_i^n$, $\boldsymbol{\mu}_i^n$ and $\Sigma_i^n$:

$$\alpha_i^n = \frac{1}{m} \sum_{j=1}^{m} P(i|\mathbf{x}_j),$$

$$\boldsymbol{\mu}_i^n = \frac{\sum_{j=1}^{m} \mathbf{x}_j P(i|\mathbf{x}_j)}{\sum_{j=1}^{m} P(i|\mathbf{x}_j)},$$

$$\Sigma_i^n = \frac{\sum_{j=1}^{m} P(i|\mathbf{x}_j)(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{m} P(i|\mathbf{x}_j)}.$$

8: Evaluate the regularized log likelihood:

$$\mathcal{L}(\Theta^n) = \sum_{i=1}^{m} \log\Big( \sum_{j=1}^{K} \alpha_j p_j(\mathbf{x}_i|\theta_j) \Big) - \lambda \sum_{i=1}^{K} \mathcal{R}_i.$$

9: **if** $\mathcal{L}(\Theta^n) < \mathcal{L}(\Theta^{n-1})$
10: $\gamma \leftarrow 0.9\gamma$, go to 6;
11: **if** $\mathcal{L}(\Theta^n) - \mathcal{L}(\Theta^{n-1}) \leq \delta$
12: break;
13: $n \leftarrow n + 1$;
14: **return** $\Theta^n$

对的

initialization. As suggested in [7], $K$-means can usually speedup up the convergence of the EM algorithm. The covariance matrices can conveniently be initialized to the sample covariances of the clusters found by the $K$-means algorithm, and the mixing coefficients can be set to the fractions of data points assigned to the respective clusters. We summarize our LapGMM algorithm in Table 1.

It would be important to note that our proposed algorithm shares some common properties with recently developed regularized clustering techniques, such as *Clustering with Local and Global Regularization* (CLGR, [45]). The CLGR algorithm not only enforces the global smoothness of the learned function, but also minimizes the label predication error at each local neighborhood. The local and global information is then unified through a regularization framework to learn a lower dimensional representation space in which either $K$-means or discretization techniques can be applied for label assignment. The major difference between CLGR and our proposed LapGMM algorithm is that CLGR is similarity-based whereas LapGMM is model-based.

### 3.4 Computational Complexity

In this section, we provide a computational complexity analysis of GMM and LapGMM. Suppose we have $m$ samples in $k$ clusters and each sample has $n$ dimensions. In the E-step of GMM (8), we need to compute the determinant of the covariance matrix which needs $O(n^3)$ operations. Thus, the E-step of GMM costs $O(kmn^3)$. In M-step, the (13), (14), and (15) cost $O(km)$, $O(km)$, and $O(kmn)$ operations, respectively. Suppose GMM converges after $t$ iterations, the computational complexity of GMM is $O(tkmn^3)$.

It is easy to see that the additional computation of LapGMM is the posterior probabilities smoothing step in

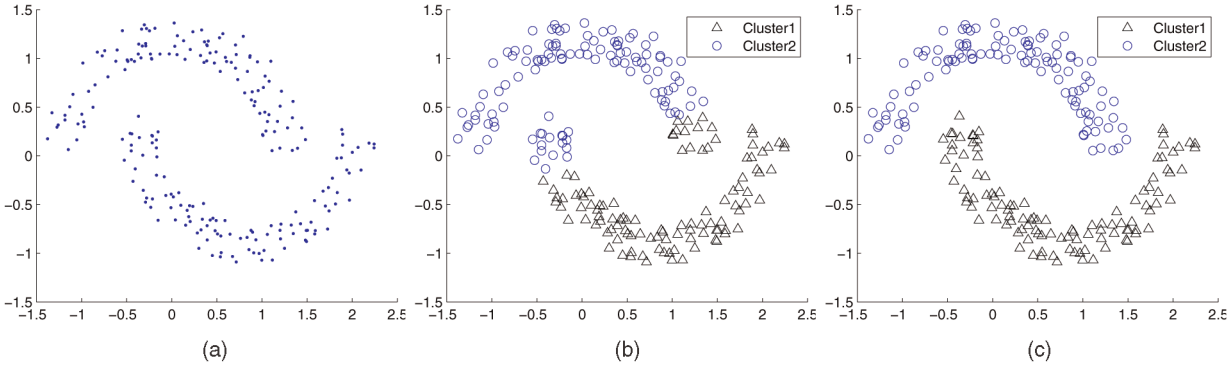



Fig. 1. Clustering on the two moons pattern. (a) Toy data set. (b) Clustering results given by GMM. (c) Clustering results given by LapGMM.

(12). Since the matrix $S$ is the weight matrix of a $p$-nearest neighbor graph, each row of $S$ has approximately $p$ nonzero elements. It is clear that (12) needs $O(kpm^2)$ operations. Finally, the computational complexity of LapGMM is $O(tkm(n^3 + pm))$.

For high-dimensional data ($n$ is large), the $n^3$ part will dominate the computation. In this case, one can use some dimensionality reduction algorithms (e.g., PCA) to reduce the dimension first.

## 4 Experimental Results

In this section, several experiments were performed to show the effectiveness of our proposed algorithm. We begin with a simple synthetic example to give some intuition about how LapGMM works.

### 4.1 A Synthetic Example

Let us consider a toy problem to explain our motivation. We are given a set of points constructed in two moons pattern, as shown in Fig. 1a. Clearly, there are two natural clusters, that is, the upper moon and the lower moon. Fig. 1b shows the clustering results obtained by the standard GMM. The two moons are mixed together. Fig. 1c shows the clustering results obtained by LapGMM. As can be seen, LapGMM yields the ideal clustering results such that the two moons are well separated. This is because LapGMM respects the local geometrical structure of the data space by incorporating a Laplacian smoothing regularizer. Therefore, nearby data points are grouped into the same cluster.

### 4.2 Data Preparation

Three real world data sets were used in our experiments. The first one is USPS handwritten digits data set,[1] which contains 9,298 images of handwritten digits. The digits 0 to 9 have 1,553, 1,269, 929, 824, 852, 716, 834, 792, 708, and 821 samples, respectively. The USPS digits data were gathered at the Center of Excellence in Document Analysis and Recognition (CEDAR) at SUNY Buffalo, as part of a project sponsored by the US postal Service. For more details about this data set, please see [27]. The size of each image is $16 \times 16$ pixels, with 256 gray levels per pixel. Thus, each image is represented by a 256-dimensional vector.

The second one is COIL20 image library[2] from Columbia. It contains 20 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. The size of each image is $32 \times 32$ pixels, with 256 gray levels per pixel. Thus, each image is represented by a 1,024-dimensional vector. Some sample images from these two data sets are shown in Figs. 2 and 3.

The third one is the TDT2 document corpus.[3] It consists of data collected during the first half of 1998 and taken from six sources, including two newswires (APW and NYT), two radio programs (VOA and PRI), and two television programs (CNN and ABC). It consists of 11,201 on-topic documents which are classified into 96 semantic categories. In this experiment, those documents appearing in two or more categories are removed, and the largest 30 categories are kept, thus leaving us with 10,021 documents in total. The number of distinct words contained in these documents is 36,771. Therefore, each document is represented as a 36,771-dimensional vector. For both GMM and LapGMM, we need to estimate the covariance matrix which is of size $36{,}771 \times 36{,}771$. However, it is practically impossible to estimate such a large matrix. On the other hand, the number of dimensions is larger than the number of data points, which suggests that the features (words) are linearly dependent. Therefore, we first apply PCA to reduce the dimension to 500.

### 4.3 Evaluation Metric

The clustering result is evaluated by comparing the obtained label of each data point with that provided by the ground truth. Two metrics, the accuracy ($AC$) and the normalized mutual information metric ($\overline{MI}$) are used to measure the clustering performance [47]. Given a point $\mathbf{x}_i$, let $r_i$ and $s_i$ be the obtained cluster label and the label provided by the ground truth, respectively. The $AC$ is defined as follows:

$$AC = \frac{\sum_{i=1}^{m} \delta(s_i, map(r_i))}{m},$$

where $m$ is the total number of samples and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise, and map($r_i$) is the permutation mapping function that maps each cluster label $r_i$ to the equivalent label

1. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps
2. http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
3. http://www.nist.gov/speech/tests/tdt/tdt98/index.htm.

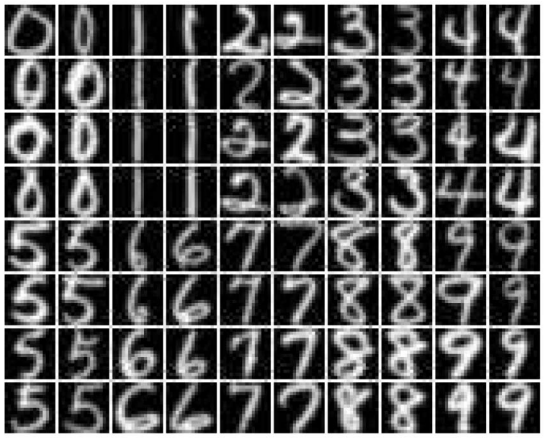


Fig. 2. Sample images from the USPS handwritten digits data set.

Fig. 3. Sample images from the COIL20 data set.

from the data set. The best mapping can be found by using the Kuhn-Munkres algorithm [36].

Let $C$ denote the set of clusters obtained from the ground truth and $C'$ obtained from our algorithm. Their mutual information metric $MI(C, C')$ is defined as follows:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)},$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a sample point arbitrarily selected from the data point belongs to the clusters $c_i$ and $c'_j$, respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected data point belongs to the clusters $c_i$ as well as $c'_j$ at the same time. In our experiments, we use the normalized mutual information $\overline{MI}$ as follows:

$$\overline{MI}(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))},$$

where $H(C)$ and $H(C')$ are the entropies of $C$ and $C'$, respectively. It is easy to check that $\overline{MI}(C, C')$ ranges from 0 to 1. $\overline{MI} = 1$ if the two sets of clusters are identical, and $\overline{MI} = 0$ if the two sets are independent.

### 4.4 Clustering Results

To demonstrate how our algorithm improves the performance of data clustering, we compared the following five methods:

- Our proposed Laplacian regularized Gaussian Mixture Model.
- $K$-means.
- Principal Component Analysis (PCA)+$K$-means.
- Nonnegative Matrix Factorization-based clustering [47].
- Gaussian Mixture Model (GMM).

$K$-means and PCA+$K$-means are considered as baseline methods. Both of them can be implemented very efficiently. GMM is the most related algorithm to our proposed method. NMF is the state-of-the-art technique for data clustering. On the other hand, GMM is a similarity-based technique, whereas NMF is a similarity-based technique for clustering. For PCA+$K$-means, we need to specify the dimensionality of the reduced subspace. In our experiments, we kept 98 percent information according to the eigenvalues. There are two important parameters in our algorithm, that is, number of nearest neighbors $p$ and regularization parameter $\lambda$. We empirically set them to 8 and $10^3$, respectively. In the next section, we will discuss the effect on clustering performance with different values of $p$ and $\lambda$. For the nearest neighbor graph in our algorithm, we use the 0-1 weighting scheme for the sake of simplicity.

Tables 2, 3, and 4 show the experimental results using the USPS, COIL20, and TDT2 data sets, respectively. The evaluations were conducted with different numbers of clusters ($k$). For USPS and COIL20, $k$ ranges from 2 to 10. For TDT2, $k$ ranges from 5 to 30. For each given cluster number $k$, 30 test runs were conducted on different

TABLE 2
Clustering Performance on USPS

| $K$ | Accuracy (%) | | | | | Normalized Mutual Information (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM |
| 2 | 95.4 | 95.4 | 92.4 | 95.5 | **98.3** | 77.0 | 77.0 | 68.9 | 78.9 | **89.8** |
| 3 | 86.3 | 86.3 | 84.8 | 87.3 | **97.1** | 71.0 | 71.0 | 67.6 | 72.9 | **90.2** |
| 4 | 79.7 | 81.3 | 73.8 | 78.4 | **90.8** | 67.7 | 67.8 | 60.3 | 69.6 | **83.8** |
| 5 | 79.2 | 79.2 | 75.0 | 79.0 | **92.6** | 68.5 | 68.5 | 63.8 | 70.9 | **87.6** |
| 6 | 79.2 | 79.2 | 74.5 | 77.1 | **86.5** | 69.0 | 69.0 | 63.1 | 70.0 | **82.5** |
| 7 | 73.6 | 74.4 | 69.1 | 71.1 | **86.9** | 65.3 | 65.2 | 60.3 | 67.6 | **83.6** |
| 8 | 72.1 | 72.9 | 68.4 | 70.4 | **84.3** | 65.7 | 65.7 | 60.7 | 68.4 | **82.9** |
| 9 | 69.9 | 71.1 | 67.3 | 68.0 | **81.2** | 65.8 | 65.8 | 60.6 | 69.1 | **82.4** |
| 10 | 69.8 | 69.8 | 67.0 | 67.3 | **79.8** | 66.3 | 66.2 | 61.7 | 69.3 | **83.0** |
| Avg | 78.4 | 78.8 | 74.7 | 77.1 | **88.6** | 68.5 | 68.5 | 63.0 | 70.7 | **85.1** |

TABLE 3
Clustering Performance on COIL20

| $K$ | Accuracy (%) | | | | | Normalized Mutual Information (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM |
| 2 | 88.4 | 88.5 | 86.2 | 88.3 | **91.7** | 68.0 | 68.1 | 60.7 | 67.1 | **77.0** |
| 3 | 83.3 | 83.5 | 75.6 | 83.4 | **89.5** | 69.0 | 69.2 | 60.4 | 69.2 | **82.2** |
| 4 | 83.0 | 83.3 | 79.1 | 83.1 | **86.8** | 74.6 | 75.2 | 71.0 | 74.7 | **82.1** |
| 5 | 77.0 | 78.9 | 74.7 | 79.6 | **85.1** | 73.7 | 75.0 | 69.7 | 75.8 | **82.4** |
| 6 | 74.5 | 77.3 | 73.4 | 77.9 | **82.9** | 73.2 | 75.3 | 71.0 | 76.4 | **82.2** |
| 7 | 70.6 | 73.2 | 68.3 | 72.4 | **74.6** | 72.0 | 73.1 | 68.2 | 74.5 | **78.3** |
| 8 | 68.6 | 71.0 | 66.8 | 66.4 | **70.4** | 71.8 | 73.8 | 70.0 | 70.5 | **77.4** |
| 9 | 70.8 | **72.9** | 69.1 | 60.1 | 68.1 | 73.7 | **74.9** | 71.8 | 65.8 | 74.3 |
| 10 | 69.6 | **72.5** | 70.0 | 58.0 | 67.8 | 75.0 | **76.2** | 73.1 | 66.5 | 75.6 |
| Avg | 76.2 | 77.9 | 73.7 | 74.4 | **79.7** | 72.3 | 73.4 | 68.4 | 71.2 | **79.1** |

TABLE 4
Clustering Performance on TDT2

| $K$ | Accuracy (%) | | | | | Normalized Mutual Information (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM | $k$-means | PCA+$k$-means | NMF | GMM | LapGMM |
| 5 | 82.5 | 89.7 | 91.5 | 86.1 | **94.0** | 79.0 | 84.1 | 85.3 | 81.7 | **87.9** |
| 10 | 66.2 | 68.2 | 74.2 | 65.6 | **81.1** | 68.7 | 70.1 | 72.7 | 69.8 | **80.9** |
| 15 | 62.4 | 63.4 | 67.2 | 60.2 | **79.8** | 72.6 | 72.4 | 73.0 | 71.3 | **79.4** |
| 20 | 59.4 | 58.7 | 62.1 | 58.4 | **80.7** | 71.1 | 70.6 | 69.4 | 71.3 | **80.8** |
| 25 | 57.6 | 57.2 | 59.5 | 54.7 | **79.7** | 71.5 | 70.5 | 67.9 | 70.0 | **81.3** |
| 30 | 58.7 | 60.9 | 62.1 | 54.0 | **79.4** | 72.1 | 72.7 | 68.9 | 69.7 | **81.6** |
| Avg. | 64.5 | 66.4 | 69.4 | 63.2 | **82.4** | 72.5 | 73.4 | 72.9 | 72.3 | **82.0** |

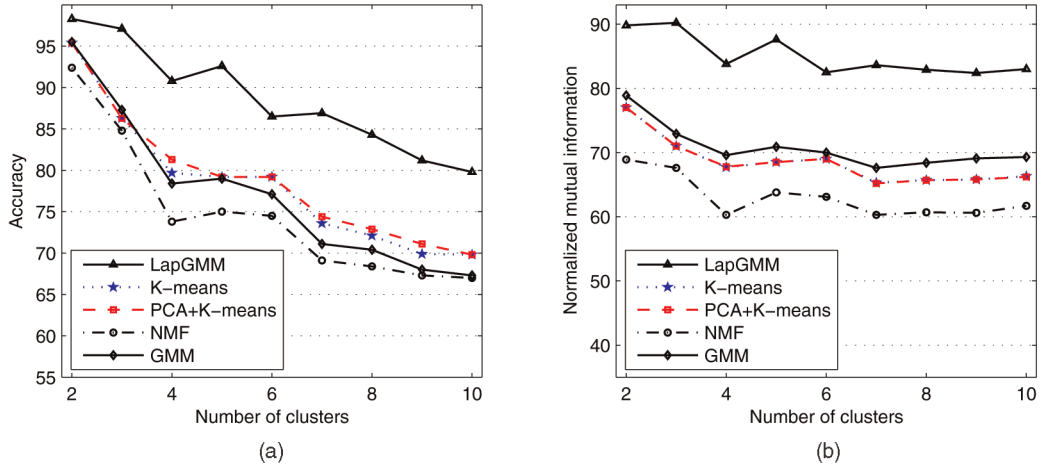



Fig. 4. (a) Accuracy. (b) Normalized mutual information versus the number of clusters on USPS data set.

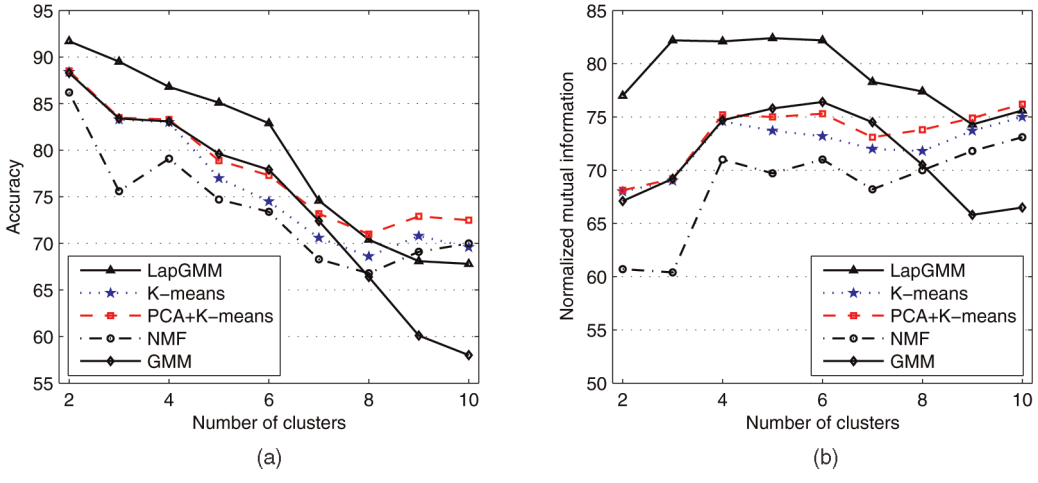



Fig. 5. (a) Accuracy. (b) Normalized mutual information versus the number of clusters on COIL20 data set.

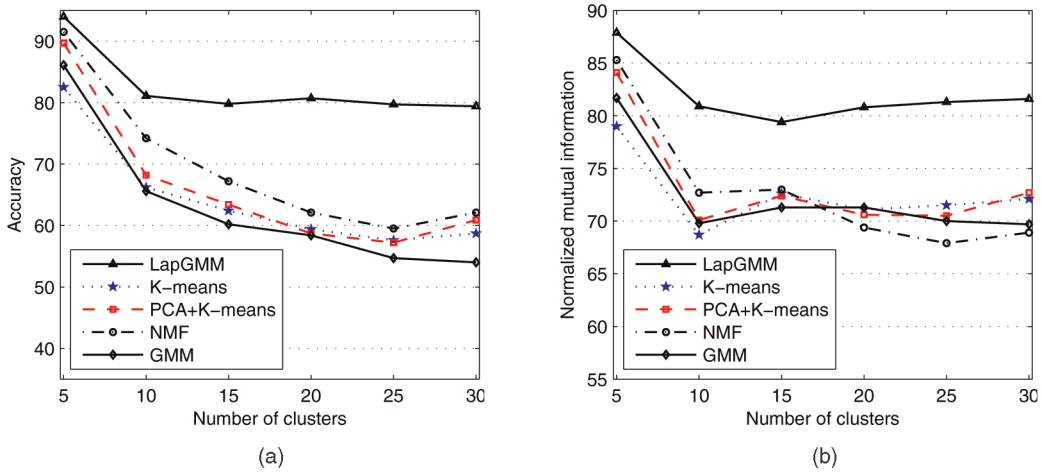



Fig. 6. (a) Accuracy. (b) Normalized mutual information versus the number of clusters on TDT2 data set.

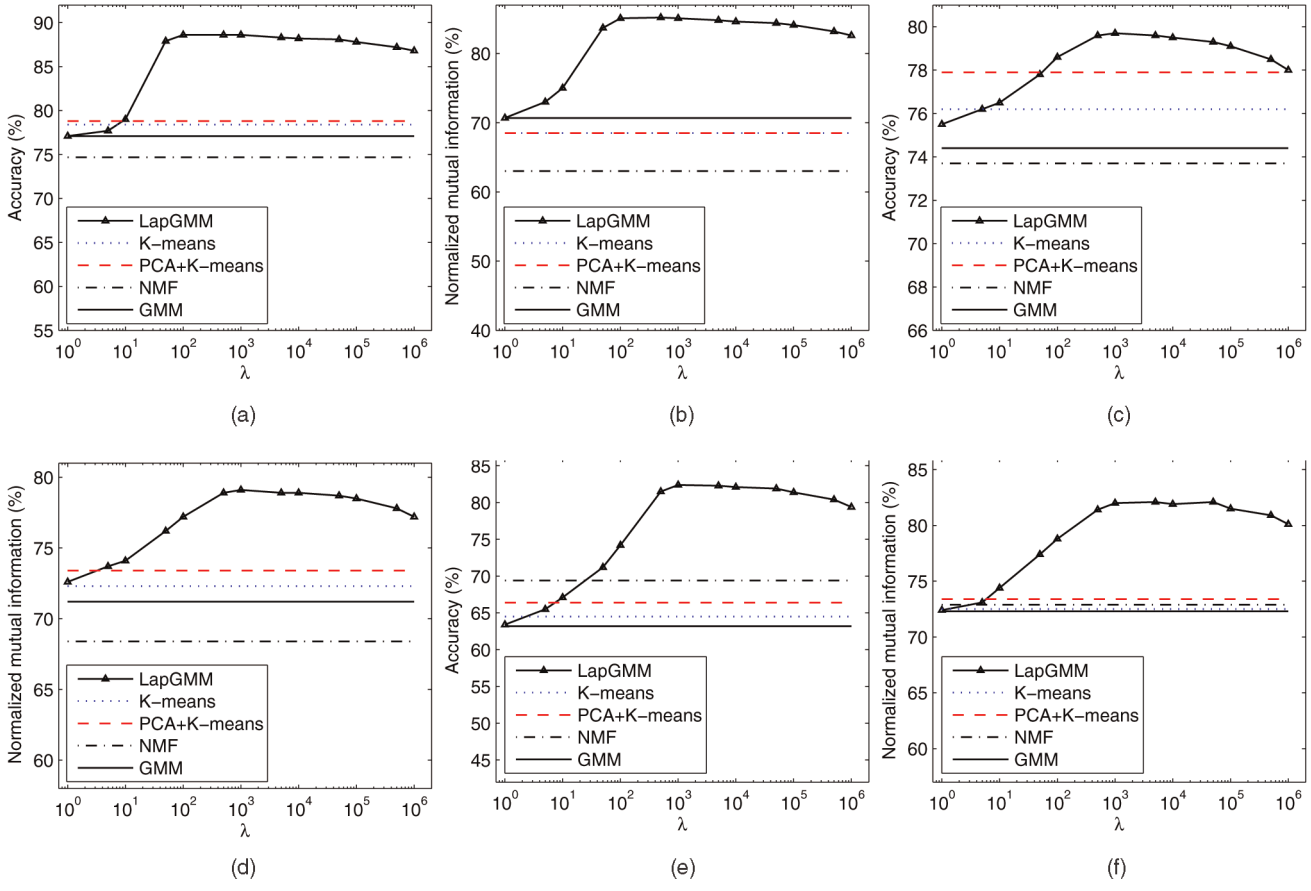



Fig. 7. The performance of LapGMM versus parameter $\lambda$. LapGMM is very stable with respect to the parameter $\lambda$. It achieves consistently good performance with the $\lambda$ varying from 500 to $10^5$. (a) USPS. (b) USPS. (c) COIL20. (d) COIL20. (e) TDT2. (f) TDT2.

randomly chosen clusters, and the final performance scores were computed by averaging the scores from the 30 tests. For each test, the $K$-means was applied 10 times with different start points and the best result in terms of the objective function of $K$-means was recorded.

As can be seen, our proposed LapGMM performs the best on all the three data sets. On the USPS and COIL20 data sets, PCA+$K$-means performs the second best, and NMF performs the worst. On TDT2, NMF performs the second best, and GMM performs the worst. On the USPS data set, the average accuracy (normalized mutual information) for LapGMM, $K$-means, PCA+$K$-means, NMF, and GMM are 88.6 (85.1 percent), 78.4 (68.5 percent), 78.8 (68.5 percent), 74.7 (63.0 percent), and 77.1 percent (70.7 percent), respectively. Our algorithm has gained 12.4 percent relative improvement in accuracy and 24.2 percent relative improvement in normalized mutual information over PCA+$K$-means. For all the cases ($K = 2, \ldots, 10$), our algorithm consistently outperforms the other four algorithms. On the COIL20 data set, the average accuracy (normalized mutual

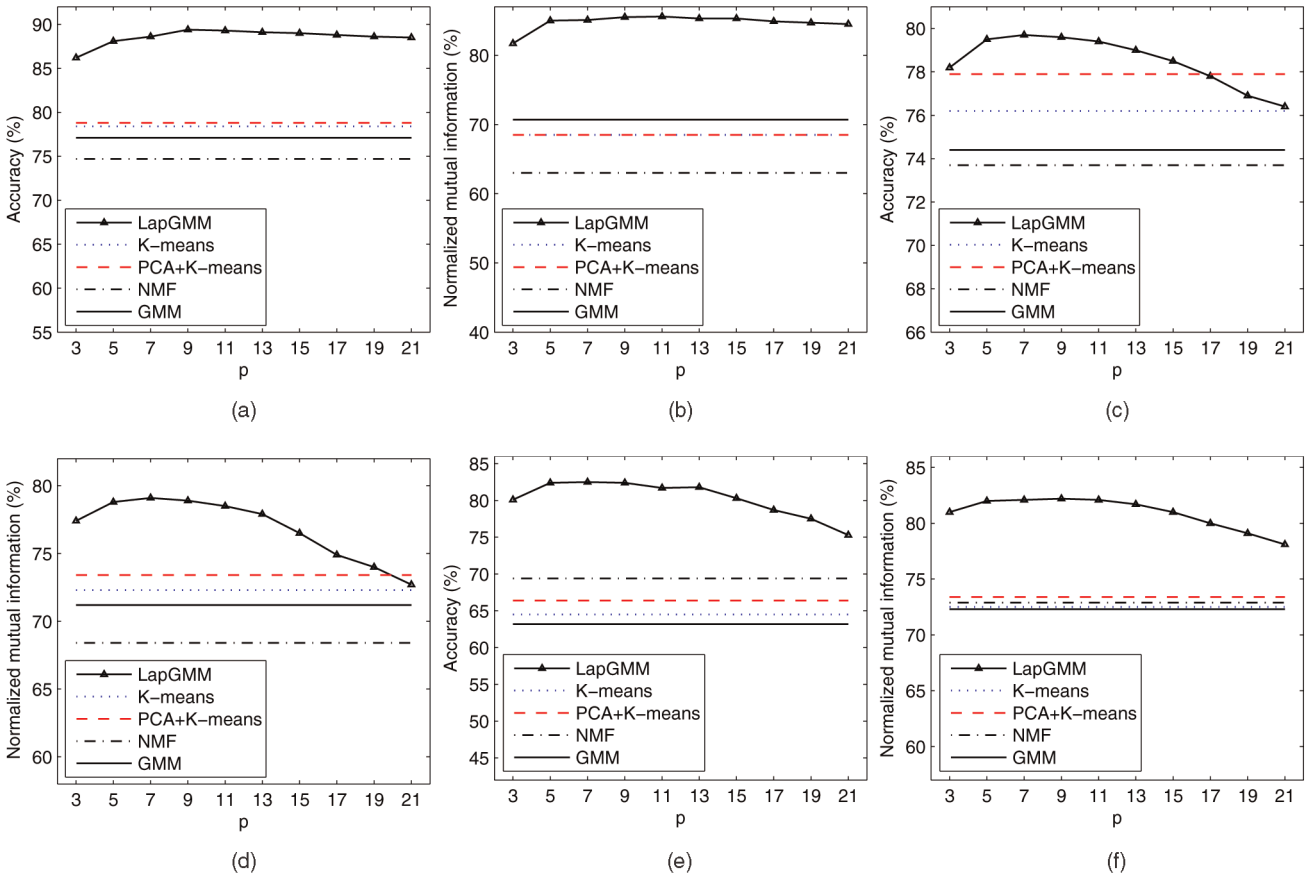



Fig. 8. The performance of LapGMM versus parameter $p$. LapGMM achieves consistently good performance with the parameter $p$ varying from 5 to 13. (a) USPS. (b) USPS. (c) COIL20. (d) COIL20. (e) TDT2. (f) TDT2.

information) for LapGMM, $K$-means, PCA+$K$-means, NMF, and GMM are 79.7 (79.1 percent), 76.2 (72.3 percent), 77.9 (73.4 percent), 73.7 (68.4 percent), and 74.4 percent (71.2 percent), respectively. On TDT2, the average accuracy (normalized mutual information) for LapGMM, $K$-means, PCA+$K$-means, NMF, and GMM are 82.4 (82.0 percent), 64.5 (72.5 percent), 66.4 (73.4 percent), 69.4 (72.9 percent), and 63.2 percent (72.3 percent), respectively. Comparing to the second best algorithm NMF, our LapGMM algorithm has achieved 18.7 and 12.5 percent relative improvement in accuracy and normalized mutual information. Figs. 4, 5, and 6 show the plots of clustering performance versus the number of clusters.

These experiments reveal a number of interesting points:

1. On all the three data sets, LapGMM outperforms the other four algorithms. As the number of clusters increases, the clustering performance for all the methods decreases.
2. LapGMM performs especially good on the USPS and TDT2 data sets. This is probably because that both USPS and TDT2 data sets contain much more samples than COIL20. Therefore, the constructed affinity graph on USPS and TDT2 data sets can better capture the intrinsic manifold structure.
3. Comparing to the standard GMM method, the LapGMM method encodes more discriminating information by smoothing the conditional probability density functions. In fact, if there is a reason to believe that Euclidean distances ($\|\mathbf{x}_i - \mathbf{x}_j\|$) are meaningful only if they are small (local), then our algorithm fits a probabilistic model that respects such a belief.

### 4.5 Model Selection

Model selection is a crucial problem in most of the learning problems. In some situations, the learning performance may drastically vary with different choices of the parameters, and we have to apply some model selection methods [44] for estimating the generalization error. In this section, we evaluate the performance of our algorithm with different values of the parameters.

Our LapGMM algorithm has two essential parameters: the number of nearest neighbors $p$ and the regularization parameter $\lambda$. $p$ defines the "locality" of the data manifold and $\lambda$ controls the smoothness of the conditional probability distributions. Figs. 7 and 8 show the performance of LapGMM as a function of the parameters $\lambda$ and $p$, respectively. As can be seen, the performance of LapGMM is very stable with respect to both of the two parameters. It achieves consistently good performance with the $\lambda$ varying from 500 to $10^5$ and $p$ varying from 5 to 13. Thus, the selection of parameters is not a very crucial problem in the LapGMM algorithm.

## 5 Conclusions and Future Work

We have introduced a novel algorithm, called Laplacian regularized Gaussian Mixture Model, for data clustering. It

is based on data-dependent geometric regularization which incorporates the local manifold structure in the log-likelihood function of the standard Gaussian mixture model. We construct a sparse nearest neighbor graph to detect the underlying nonlinear manifold structure and use the graph Laplacian to smooth the conditional probability density function along the geodesics in the intrinsic geometry of the marginal distribution. Experimental results on USPS handwritten digits data set, COIL20 image library, and TDT2 document corpus show the effectiveness of our method.

Several questions remain to be investigated in our future work.

1. Central to the algorithm is a graph structure that is inferred on the data points. Recently there have been a lot of interest in exploring different ways of constructing a graph to model the intrinsic geometrical and discriminative structure in the data [14], [3]. There is no reason to believe that the nearest neighbor graph is the only or the most natural choice. Also, as we described above, graph Laplacian is not the only choice of smoothing operators. A systematic investigation of the use of different graph-based smoothing operators needs to be carried out.
2. Our approach utilizes the local metric structure of the manifold to improve data clustering. We have not given any consideration to other topological invariants of the manifold that may be potentially estimated from data. For example, it remains unclear how to reliably estimate the number of connected components (or, clusters).
3. There are several parameters in our algorithm, e.g., number of nearest neighbors and the regularization parameter. Although our empirical evaluation shows that our algorithm is not sensitive to the parameters, it remains unclear how to do model selection in a principled manner. The major difficulty of model selection for clustering is due to the lack of label information.

## ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China under Grant 60875044, National Key Basic Research Foundation of China under Grant 2009CB320801, and the US National Science Foundation (NSF) under Grants IIS-08-42769 and IIS-09-05215.

## REFERENCES

[1] Z. Abbassi and V.S. Mirrokni, "A Recommender System Based on Local Random Walks and Spectral Methods," *Proc. Ninth WebKDD and First SNA-KDD Workshop Web Mining and Social Network Analysis,* 2007.

[2] C.C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. Zaki, "Xproj: A Framework for Projected Structural Clustering of XML Documents," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 46-55, 2007.

[3] A. Argyriou, M. Herbster, and M. Pontil, "Combining Graph Laplacian for Semi-Supervised Learning," *Advances in Neural Information Processing Systems 18,* MIT Press, 2005.

[4] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant, "Robust Information-Theoretic Clustering," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 65-75, Aug. 2006.

[5] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," *Advances in Neural Information Processing Systems 14,* pp. 585-591, MIT Press, 2001.

[6] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Examples," *J. Machine Learning Research,* vol. 7, pp. 2399-2434, 2006.

[7] C.M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2007.

[8] J.-P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov, "Metagenes and Molecular Pattern Discovery Using Matrix Factorization," *Proc. Nat'l Academy of Sciences,* vol. 101, no. 12, pp. 4164-4169, 2004.

[9] D. Cai and X. He, "Orthogonal Locality Preserving Indexing," *Proc. ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '05),* pp. 3-10, 2005.

[10] D. Cai, X. He, and J. Han, "Document Clustering Using Locality Preserving Indexing," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 12, pp. 1624-1637, Dec. 2005.

[11] D. Cai, X. He, W.-Y. Ma, J.-R. Wen, and H.-J. Zhang, "Organizing www Images Based on the Analysis of Page Layout and Web Link Structure," *Proc. IEEE Int'l Conf. Multimedia and Expo (ICME '04),* 2004.

[12] D. Cai, Q. Mei, J. Han, and C. Zhai, "Modeling Hidden Topics on Document Manifold," *Proc. ACM Int'l Conf. Information and Knowledge Management,* 2008.

[13] D. Cai, X. Wang, and X. He, "Probabilistic Dyadic Data Analysis with Local and Global Consistency," *Proc. 26th Int'l Conf. Machine Learning,* 2009.

[14] M.A. Carreira-Perpinan and R.S. Zemel, "Proximity Graphs for Clustering and Manifold Learning," *Proc. Advances in Neural Information Processing Systems 17,* MIT Press, Dec. 2004.

[15] E. Cesario, G. Manco, and R. Ortale, "Top-Down Parameter-Free Clustering of High-Dimensional Categorical Data," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 12, pp. 1607-1624, Dec. 2007.

[16] P.K. Chan, D.F. Schlag, and J.Y. Zien, "Spectral k-Way Ratio-Cut Partitioning and Clustering," *IEEE Trans. Computer-Aided Design,* vol. 13, no. 9, pp. 1088-1096, Sept. 1994.

[17] M. Chu, F. Diele, R. Plemmons, and S. Ragni, "Optimality, Computation, and Interpretation of Nonnegative Matrix Factoriaztions," *SIAM J. Matrix Analysis,* Preprint, http://www.wfu.edu/~plemmons/papers/chu_ple.pdf.

[18] F.R.K. Chung, *Spectral Graph Theory, Volume 92 of Regional Conf. Series in Mathematics.* AMS, 1997.

[19] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Self-Taught Clustering," *Proc. 25nd Int'l Conf. Machine Learning,* July 2008.

[20] I. Davidson, M. Ester, and S.S. Ravi, "Efficient Incremental Constraint Clustering," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 240-249, Aug. 2007.

[21] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the Em Algorithm," *J. Royal Statistical Soc., Series B (Methodological),* vol. 39, no. 1, pp. 1-38, 1977.

[22] C. Ding, T. Li, and M. Jordan, "Convex and Semi-Nonnegative Matrix Factorizations for Clustering and Low-Dimension Representation," Technical Report LBNL-60428, Lawrence Berkeley Nat'l Laboratory, 2006.

[23] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 126-135, 2006.

[24] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification,* second ed. Wiley-Interscience, 2000.

[25] D.J. Higham, G. Kalna, and M. Kibble, "Spectral Clustering and Its Use in Bioinformatics," *J. Computational and Applied Math.,* vol. 204, no. 1, pp. 25-37, 2007.

[26] P.O. Hoyer, "Non-Negative Matrix Factorizaiton with Sparseness Constraints," *J. Machine Learning Research,* vol. 5, pp. 1457-1469, 2004.

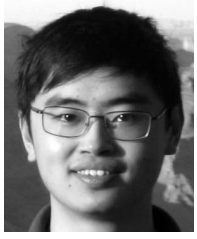
[27] J.J. Hull, "A Database for Handwritten Text Recognition Research," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 5, pp. 550-554, May 1994.

[28] C.S. Jensen, "Continuous Clustering of Moving Objects," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 9, pp. 1161-1174, Sept. 2007.

[29] N. Kumar and K. Kummamuru, "Semisupervised Clustering with Metric Learning Using Relative Comparisons," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 4, pp. 496-503, Apr. 2008.

[30] D.D. Lee and H.S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature,* vol. 401, pp. 788-791, 1999.

[31] D.D. Lee and H.S. Seung, "Algorithms for Non-Negative Matrix Factorization," *Proc. Advances in Neural Information Processing Systems 13,* MIT press, 2001.
[32] J.M. Lee, *Introduction to Smooth Manifolds.* Springer-Verlag, 2002.
[33] S.Z. Li, X. Hou, H. Zhang, and Q. Cheng, "Learning Spatially Localized, Parts-Based Representation," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition (CVPR '01),* pp. 207-212, 2001.
[34] T. Li and C. Ding, "The Relationships among Various Nonnegative Matrix Factorization Methods for Clustering," *Proc. Int'l Conf. Data Mining (ICDM '06),* 2006.
[35] X. Li, S. Lin, S. Yan, and D. Xu, "Discriminant Locally Linear Embedding with High-Order Tensor Data," *IEEE Trans. Systems, Man, and Cybernetics, Part B,* vol. 38, no. 2, pp. 342-352, Apr. 2008.
[36] L. Lovasz and M. Plummer, *Matching Theory.* Akadémiai Kiadó, 1986.
[37] Q. Mei, D. Cai, D. Zhang, and C. Zhai, "Topic Modeling with Network Regularization," *Proc. 17th Int'l World Wide Web Conf.,* 2008.
[38] R. Neal and G. Hinton, "A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants," *Learning in Graphical Models,* Kluwer, 1998.
[39] A.Y. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems 14,* pp. 849-856, MIT Press, 2001.
[40] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, no. 5500, pp. 2323-2326, 2000.
[41] H.S. Seung and D.D. Lee, "The Manifold Ways of Perception," *Science,* vol. 290, no. 12, pp. 2268-2269, 2000.
[42] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[43] J. Tenenbaum, V. de Silva, and J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, no. 5500, pp. 2319-2323, Aug. 2000.
[44] V.N. Vapnik, *Statistical Learning Theory.* John Wiley and Sons, 1998.
[45] F. Wang, C. Zhang, and T. Li, "Regularized Clustering for Documents," *Proc. Int'l Conf. Research and Development in Information Retrieval (SIGIR '07),* July 2007.
[46] W. Xu and Y. Gong, "Document Clustering by Concept Factorization," *Proc. ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '04),* pp. 202-209, July 2004.
[47] W. Xu, X. Liu, and Y. Gong, "Document Clustering Based on Non-Negative Matrix Factorization," *Proc. Int'l Conf. Research and Development in Information Retrieval (SIGIR '03),* pp. 267-273, Aug. 2003.
[48] Z. Zhang and H. Zha, "Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment," *SIAM J. Scientific Computing,* vol. 26, no. 1, pp. 313-338, 2004.
[49] X. Zhu and J. Lafferty, "Harmonic Mixtures: Combining Mixture Models and Graph-Based Methods for Inductive and Scalable Semi-Supervised Learning," *ICML '05: Proc. 22nd Int'l Conf. Machine Learning,* pp. 1052-1059, 2005.



**Xiaofei He** received the BS degree in computer science from Zhejiang University, China, in 2000 and the PhD degree in computer science from the University of Chicago, in 2005. He is a professor in the State Key Lab of CAD&CG at Zhejiang University, China. Prior to joining Zhejiang University, he was a Research Scientist at Yahoo! Research Labs, Burbank, CA. His research interests include machine learning, information retrieval, and computer vision. He is a member of the IEEE.

**Deng Cai** received the PhD degree in computer science from the University of Illinois at Urbana Champaign in 2009. Currently he is working as an associate professor in the State Key Lab of CAD&CG, College of Computer Science at Zhejiang University, China. His research interests are machine learning, data mining, and information retrieval.

**Yuanlong Shao** received his BS degree in computer science from Zhejiang University, China, in 2007. Currently, he is working toward the master's degree in computer science at State Key Lab of CAD&CG, Zhejiang University. His research interests include machine learning, manifold learning, probabilistic inference, and image annotation.

**Hujun Bao** received his bachelor and PhD degrees in applied mathematics from Zhejiang university in 1987 and 1993. His research interests include computer graphics, computer vision, and virtual reality. Currently, he is the director of State Key Lab of CAD&CG of Zhejiang university, China.

**Jiawei Han** is a professor in the Department of Computer Science at the University of Illinois. He has been working on research into data mining, data warehousing, stream data mining, spatiotemporal and multimedia data mining, information network analysis, text and Web mining, and software bug mining, with more than 400 conference and journal publications. He has chaired or served in more than 100 program committees of international conferences and workshops and also served or is serving on the editorial boards for *Data Mining and Knowledge Discovery, IEEE Transactions on Knowledge and Data Engineering, Journal of Computer Science and Technology*, and *Journal of Intelligent Information Systems*. Currently, he is the founding editor-in-chief of *ACM Transactions on Knowledge Discovery from Data* (*TKDD*). He has received IBM Faculty Awards, the Outstanding Contribution Award at the International Conference on Data Mining (2002), ACM SIGKDD Innovation Award (2004), and IEEE Computer Society Technical Achievement Award (2005). His book "*Data Mining: Concepts and Techniques*" (Morgan Kaufmann) has been used worldwide as a textbook. He is a Fellow of the ACM and the IEEE.