IEICE TRANS. INF. & SYST., VOL.E102–D, NO.6 JUNE 2019

PAPER

Direct Log-Density Gradient Estimation with Gaussian Mixture Models and Its Application to Clustering

Qi ZHANG[†a)], **Hiroaki SASAKI**[††b)], *Nonmembers*, *and* **Kazushi IKEDA**[††c)], *Senior Member*

SUMMARY Estimation of the gradient of the logarithm of a probability density function is a versatile tool in statistical data analysis. A recent method for model-seeking clustering called the *least-squares log-density gradient clustering* (LSLDGC) [Sasaki et al., 2014] employs a sophisticated gradient estimator, which directly estimates the log-density gradients without going through density estimation. However, the typical implementation of LSLDGC is based on a spherical Gaussian function, which may not work well when the probability density function for data has highly correlated local structures. To cope with this problem, we propose a new gradient estimator for log-density gradients with Gaussian mixture models (GMMs). Covariance matrices in GMMs enable the new estimator to capture the highly correlated structures. Through the application of the new gradient estimator to mode-seeking clustering and hierarchical clustering, we experimentally demonstrate the usefulness of our clustering methods over existing methods.

key words: *probability density gradient, mixture model, clustering, hierarchical clustering*

1. Introduction

Estimating the gradient of the logarithm of the probability density function underlying data is an important problem. For instance, an unsupervised dimensionality reduction method requires to estimate the log-density gradient [1]. In a supervised learning problem, estimating the log-density gradient enables us to capture multiple functional relationship between output and input variables [2]. Other statistical topics can be seen in [3]. Thus, log-density gradient estimation offers a certain range of applications in statistical data analysis.

Among them, an interesting application is mode-seeking clustering. *Mean shift clustering* (MS) [4]–[6] iteratively updates data samples toward the modes (i.e., local maxima) of the estimated probability density function by gradient ascent, and then assigns a cluster label to the data samples which converged to the same mode. Compared with k-means and mixture-model-based clustering [7], MS has two notable points: MS does not make strong assumptions on the probability density function and the number of clusters in MS is automatically determined by the detected modes. Therefore, MS has been applied to a variety of problems such as image segmentation [6], [8], [9] and object tracking [10], [11] (See also a recent review article [12]). Furthermore, MS and its related methods have been extended for handling manifold data [13], [14] and to hierarchical clustering [15].

A native approach is first to estimate the probability density function (e.g., by kernel density estimation), and then to compute its log-gradient. However, this approach can be unreliable because a good density estimator does not necessarily mean a good gradient estimator. To alleviate this problem, a recent mode-seeking clustering method called *LSLDG clustering* employs a sophisticated gradient estimator, which *directly* fits a gradient model to the true log-density gradient without going through density estimation [16]–[18]. LSLDG clustering has been experimentally demonstrated to significantly improve the performance of MS particularly for high-dimensional data [17], [18]. However, the gradient estimator in LSLDG clustering is typically implemented based on the spherical Gaussian kernel. Thus, when the probability density function includes clusters with highly correlated structures, LSLDG may require a huge number of samples to produce a smooth gradient estimate. This can be problematic particularly in mode-seeking clustering because an unsmooth estimate may create spurious modes as we demonstrate later.

To improve the performance of LSLDG clustering to the highly correlated structures, we propose to use a Gaussian mixture model (GMM) in log-density gradient estimation. Estimating the covariance matrices in GMM makes the gradient estimator much more adaptive to the local structures in the probability density function. A challenge is to satisfy the positive semidefinite constraint of the covariance matrices. To overcome this challenge, we develop an estimation algorithm combined with manifold optimization [19].

Next, we apply the proposed estimator to mode-seeking clustering. To update data samples during mode-seeking, we derive an update formula based on the fixed-point method. A similar formula has been proposed in MS [20], but as shown later, our formula includes it as a special case. Furthermore, we extend the proposed clustering method to hierarchical clustering, which is a novel extension of LSLDG clustering. The usefulness of the proposed clustering methods are experimentally demonstrated.

This paper is organized as follows: Sect. 2 reviews an existing gradient estimator and MS. In Sect. 3, we propose a

Manuscript received October 20, 2018.
Manuscript revised February 14, 2019.
Manuscript publicized March 22, 2019.

[†]The author is with the Graduate University for Advanced Studies, Tachikawa-shi, 190–0014 Japan.
[††]The authors are with Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: qiz@ism.ac.jp
b) E-mail: hsasaki@is.naist.jp
c) E-mail: kazushi@is.naist.jp

DOI: 10.1587/transinf.2018EDP7354

new estimator for log-density gradients with Gaussian mixture models. Section 4 develops a mode-seeking clustering method as an application of the proposed gradient estimator. Then, the developed clustering method is further extended to hierarchical clustering. The performance of the clustering methods are experimentally investigated in Sect. 5. Section 6 concludes this paper.

2. Background

This section reviews an existing estimator for log-density gradients and mode-seeking clustering methods.

2.1 Review of LSLDG

Here, we review a direct estimator for log-density gradients which we refer to the *least-squares log-density gradients* (LSLDG) [16], [17]. Suppose that n i.i.d. data samples drawn from a probability distribution with density $p(\boldsymbol{x})$ are available as

$$\mathcal{D} := \{\boldsymbol{x}_i = (x_i^{(1)}, \ldots, x_i^{(d)})\}_{i=1}^n \sim p(\boldsymbol{x}).$$

The goal of LSLDG is to estimate the gradient of the log-density from $\mathcal{D}$:

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) = \left(\frac{\partial_1 p(\boldsymbol{x})}{p(\boldsymbol{x})}, \ldots, \frac{\partial_d p(\boldsymbol{x})}{p(\boldsymbol{x})}\right)^\top,$$

where $\partial_j := \frac{\partial}{\partial x^{(j)}}$ and $\nabla_{\boldsymbol{x}}$ denotes the differential operator with respect to $\boldsymbol{x}$.

LSLDG directly fits a model g_j to the true partial derivative of the log-density under the squared-loss:

$$\begin{aligned} J_j(g_j) &:= \frac{1}{2}\int \{g_j(\boldsymbol{x}) - \partial_j \log p(\boldsymbol{x})\}^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - C_j \\ &= \frac{1}{2}\int \{g_j(\boldsymbol{x})\}^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int g_j(\boldsymbol{x})\left\{\partial_j p(\boldsymbol{x})\right\}\mathrm{d}\boldsymbol{x} \\ &= \frac{1}{2}\int \{g_j(\boldsymbol{x})\}^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \int \left\{\partial_j g_j(\boldsymbol{x})\right\} p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \end{aligned}$$

where $C_j := \frac{1}{2}\int \{\partial_j \log p(\boldsymbol{x})\}^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, and we applied the *integration by parts* to the second term on the right-hand side under the assumption that $|g_j(\boldsymbol{x})p(\boldsymbol{x})| \to 0$ as $|x_j| \to \infty$. Then, the empirical risk up to the ignorable constant C_j is obtained as

$$\widehat{J}_j(g_j) := \frac{1}{n}\sum_{i=1}^n \left[\frac{1}{2} g_j(\boldsymbol{x}_i)^2 + \partial_j g_j(\boldsymbol{x}_i)\right]. \tag{1}$$

To estimate g_j, LSLDG employs the following model:

$$g_j(\boldsymbol{x}) := \sum_{i=1}^b \alpha_i^{(j)} \varphi_i^{(j)}(\boldsymbol{x}), \tag{2}$$

where $\alpha_i^{(j)}$ and $\varphi_i^{(j)}$ are coefficients and basis functions respectively, and b denotes the number of basis functions. In [17], the derivative of the Gaussian kernel is used as

$$\varphi_i^{(j)}(\boldsymbol{x}) := \partial_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_j^2}\right), \tag{3}$$

where $\boldsymbol{c}_i := (c_i^{(1)}, \ldots, c_i^{(d)})^\top$ is the kernel centers fixed at a subset of data samples randomly chosen from $\mathcal{D}$, and σ_j denotes the bandwidth parameter. After substituting the model (2) into $\widehat{J}_j$, the optimal $\boldsymbol{\alpha}^{(j)} := (\alpha_1^{(j)}, \ldots, \alpha_b^{(j)})^\top$ can be computed analytically by solving the following problem:

$$\begin{aligned} \widehat{\boldsymbol{\alpha}}^{(j)} &:= \operatorname*{argmin}_{\boldsymbol{\alpha}^{(j)}} \widehat{J}_j(\boldsymbol{\alpha}^{(j)}) + \lambda_j \|\boldsymbol{\alpha}^{(j)}\|^2 \\ &= -(\widehat{\boldsymbol{G}}_j + \lambda_j \boldsymbol{I}_b)^{-1}\widehat{\boldsymbol{h}}_j, \end{aligned}$$

where λ_j is the regularization parameter, $\boldsymbol{I}_b$ denotes the b by b identity matrix, and with $\boldsymbol{\varphi}^{(j)}(\boldsymbol{x}) := (\varphi_1^{(j)}(\boldsymbol{x}), \ldots, \varphi_b^{(j)}(\boldsymbol{x}))^\top$,

$$\widehat{\boldsymbol{G}}_j := \frac{1}{n}\sum_{i=1}^n \boldsymbol{\varphi}^{(j)}(\boldsymbol{x}_i)\boldsymbol{\varphi}^{(j)}(\boldsymbol{x}_i)^\top, \quad \widehat{\boldsymbol{h}}_j := \frac{1}{n}\sum_{i=1}^n \partial_j \boldsymbol{\varphi}^{(j)}(\boldsymbol{x}_i).$$

The gradient estimator is finally given by

$$\widehat{g}_j(\boldsymbol{x}) = \sum_{i=1}^b \widehat{\alpha}_i^{(j)} \varphi_i^{(j)}(\boldsymbol{x}).$$

2.2 Review of Mode-Seeking Clustering

Mean shift clustering (MS) [4], [6] is a mode-seeking clustering method based on *kernel density estimation* (KDE). MS employs KDE to estimate the probability density function as follows:

$$\widehat{p}_{\mathrm{KDE}}(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^n K\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right), \tag{4}$$

where h denotes the bandwidth parameter and K is a smooth kernel function for KDE (See [6, Eq. (3)] for definition). Taking the gradient of $\widehat{p}_{\mathrm{KDE}}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ yields

$$\begin{aligned} &\nabla_{\boldsymbol{x}} \widehat{p}_{\mathrm{KDE}}(\boldsymbol{x}) \\ &= \frac{1}{n}\sum_{i=1}^n \frac{\boldsymbol{x}_i - \boldsymbol{x}}{h^2} H\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right) \\ &= \frac{1}{nh^2}\left[\sum_{i=1}^n \boldsymbol{x}_i H\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right) - \boldsymbol{x}\sum_{i=1}^n H\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right)\right] \end{aligned}$$

where $H(t) := -\frac{\mathrm{d}}{\mathrm{d}t}K(t)$. Based on the fixed-point method, setting $\nabla_{\boldsymbol{x}} \widehat{p}_{\mathrm{KDE}}(\boldsymbol{x}) = \boldsymbol{0}$ yields a simple update formula as

$$\boldsymbol{x} \leftarrow \frac{\sum_{i=1}^n \boldsymbol{x}_i H\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right)}{\sum_{i=1}^n H\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2h^2}\right)}.$$

This update formula is iteratively applied for data samples $\boldsymbol{x}_i$ until they converge to the modes of $\widehat{p}_{\mathrm{KDE}}$. MS finally assigns the same cluster label to the data samples converging to the same mode.

LSLDG clustering [17], [18] is another mode-seeking clustering method. The main difference is that $\nabla_{\boldsymbol{x}} \widehat{p}_{\mathrm{KDE}}(\boldsymbol{x})$

is replaced with the LSLDG estimator $\widehat{\boldsymbol{g}}(\boldsymbol{x}) = (\widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_d(\boldsymbol{x}))^\top$. An update formula is derived based on the fixed-point method in LSLDG clustering as well. It has been experimentally demonstrated that LSLDG clustering significantly outperforms MS for high-dimensional data.

3. LSLDG with Gaussian Mixture Models

To improve the performance of LSLDG to local correlation structures in the probability density function, instead of the spherical Gaussian kernel in (3), we use a Gaussian function with the mean parameter $\boldsymbol{\mu}_i$ and precision matrix $\boldsymbol{\Lambda}_i$ (i.e., the inverse of a covariance matrix) as the basis function:

$$\begin{aligned}\psi_i^{(j)}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) &:= \partial_j \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Lambda}_i (\boldsymbol{x}-\boldsymbol{\mu}_i)\right\} \\ &= [\boldsymbol{\Lambda}_i(\boldsymbol{\mu}_i - \boldsymbol{x})]_j \phi_i(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i), \end{aligned} \tag{5}$$

where $[\boldsymbol{x}]_j$ denotes the j-th element in $\boldsymbol{x}$ and

$$\phi_i(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) := \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Lambda}_i (\boldsymbol{x}-\boldsymbol{\mu}_i)\right\}.$$

Then, a gradient model based on a Gaussian mixture model[†] is given by

$$g_j^{\mathrm{GM}}(\boldsymbol{x}) := \sum_{i=1}^b \theta_i^{(j)} \psi_i^{(j)}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i), \tag{6}$$

where $\theta_i^{(j)}$ denote coefficients. Substituting the Gaussian mixture model (6) into $\widehat{J}_j$ yields the following optimization problem:

$$\begin{aligned}&\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^d, \{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^b, \{\widehat{\boldsymbol{\Lambda}}_i\}_{i=1}^b \\ &:= \operatorname*{argmin}_{\{\boldsymbol{\theta}^{(j)}\}_{j=1}^d, \{\boldsymbol{\mu}_i\}_{i=1}^b, \{\boldsymbol{\Lambda}_i\}_{i=1}^b} \sum_{j=1}^d \widehat{J}_j(\boldsymbol{\theta}^{(j)}, \{\boldsymbol{\mu}_i\}_{i=1}^b, \{\boldsymbol{\Lambda}_i\}_{i=1}^b) \\ &\qquad \text{subject to } \boldsymbol{\Lambda}_i = \boldsymbol{\Lambda}_i^\top,\ \boldsymbol{\Lambda}_i \succeq \mathbf{O} \text{ for all } i, \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}^{(j)} := (\theta_1^{(j)}, \ldots, \theta_b^{(j)})^\top$, and $\boldsymbol{\Lambda}_i \succeq \mathbf{O}$ means that $\boldsymbol{\Lambda}_i$ is positive semidefinite.

We next describe our estimation algorithm. Our algorithm is to alternately repeat two steps until $\sum_{j=1}^d \widehat{J}_j$ converges: (1) Optimization of $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^d$ and (2) of $\{\boldsymbol{\mu}_i\}_{i=1}^b$ and $\{\boldsymbol{\Lambda}_i\}_{i=1}^b$. The details of the two steps are given as follows:

(1) Optimization of $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^d$

Given $\{\boldsymbol{\mu}_i\}_{i=1}^b$ and $\{\boldsymbol{\Lambda}_i\}_{i=1}^b$, the optimal $\boldsymbol{\theta}^{(j)}$ for all j can be computed analytically[††]:

$$\widehat{\boldsymbol{\theta}}^{(j)} = \boldsymbol{G}_j^{-1} \boldsymbol{h}_j, \tag{8}$$

where with $\boldsymbol{\psi}^{(j)}(\boldsymbol{x}) = (\psi_1^{(j)}(\boldsymbol{x}), \ldots, \psi_b^{(j)}(\boldsymbol{x}))^\top$,

$$\boldsymbol{G}_j := \frac{1}{n}\sum_{i=1}^n \boldsymbol{\psi}^{(j)}(\boldsymbol{x}_i)\boldsymbol{\psi}^{(j)}(\boldsymbol{x}_i)^\top, \quad \boldsymbol{h}_j := \frac{1}{n}\sum_{i=1}^n \partial_j \boldsymbol{\psi}^{(j)}(\boldsymbol{x}_i).$$

(2) Optimization of $\{\boldsymbol{\mu}_i\}_{i=1}^b$ and $\{\boldsymbol{\Lambda}_i\}_{i=1}^b$

Given $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^d$, we optimize $\{\boldsymbol{\mu}_i\}_{i=1}^b$ and $\{\boldsymbol{\Lambda}_i\}_{i=1}^b$. Here, the challenge is that $\boldsymbol{\Lambda}_i$ must be symmetric positive semidefinite. To satisfy the constraint, we employ manifold optimization techniques [19]. In this step, gradient descent on a manifold is performed by a single step (iteration)[†††]. Regarding $\boldsymbol{\mu}_i$, the standard gradient descent is applied with a single step.

After repeating the alternate optimization over the parameters, we finally obtain the estimator as

$$\widehat{g}_j^{\mathrm{GM}}(\boldsymbol{x}) := \sum_{i=1}^b \widehat{\theta}_i^{(j)} \psi_i^{(j)}(\boldsymbol{x}; \widehat{\boldsymbol{\mu}}_i, \widehat{\boldsymbol{\Lambda}}_i). \tag{9}$$

We call the estimator the *Gaussian mixture LSLDG* (GM-LSLDG).

4. Application to Clustering

This section applies GM-LSLDG to mode-seeking clustering. Furthermore, the clustering method is extended to hierarchical clustering.

4.1 Mode-Seeking Clustering

Here, we derive an update formula of data samples to perform mode-seeking as done in MS and LSLDG clustering. Let us denote the estimated gradient vector by GM-LSLDG as $\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}) := (\widehat{g}_1^{\mathrm{GM}}(\boldsymbol{x}), \ldots, \widehat{g}_d^{\mathrm{GM}}(\boldsymbol{x}))^\top$. Then, by expanding the right-hand side of (9), $\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x})$ can be expressed as

$$\begin{aligned}\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}) &= \sum_{i=1}^b \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i (\widehat{\boldsymbol{\mu}}_i - \boldsymbol{x}) \widehat{\phi}_i(\boldsymbol{x}) \\ &= \sum_{i=1}^b \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\boldsymbol{\mu}}_i \widehat{\phi}_i(\boldsymbol{x}) - \left\{\sum_{i=1}^b \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i(\boldsymbol{x})\right\} \boldsymbol{x}, \end{aligned} \tag{10}$$

where $\widehat{\phi}_i^{(j)}(\boldsymbol{x}) := \phi_i^{(j)}(\boldsymbol{x}; \widehat{\boldsymbol{\mu}}_i, \widehat{\boldsymbol{\Lambda}}_i)$ and $\boldsymbol{\Theta}_i$ denotes the d by d diagonal matrix with diagonals $\widehat{\theta}_i^{(1)}, \ldots, \widehat{\theta}_i^{(d)}$ for a fixed i. Based on the fixed-point method, setting $\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}) = \mathbf{0}$ yields the following update formula:

$$\boldsymbol{x} \leftarrow \left\{\sum_{i=1}^b \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i(\boldsymbol{x})\right\}^{-1} \left\{\sum_{i=1}^b \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\boldsymbol{\mu}}_i \widehat{\phi}_i(\boldsymbol{x})\right\}. \tag{11}$$

A similar update formula has been derived in MS with a

[†]This model is different from the standard Gaussian mixture model in statistics because the Gaussian functions in (6) are not normalized. Nonetheless, we call it a Gaussian mixture model to emphasize that the mean parameter $\boldsymbol{\mu}_i$ and precision matrix $\boldsymbol{\Lambda}_i$ are estimated from data samples.

[††]The analytic solution can be easily derived by solving $\nabla_{\boldsymbol{\theta}^{(j)}} J_j(\boldsymbol{\theta}^{(j)}, \{\boldsymbol{\mu}_i\}_{i=1}^b, \{\boldsymbol{\Lambda}_i\}_{i=1}^b) = \mathbf{0}$ with respect to $\boldsymbol{\theta}^{(j)}$.

[†††]We used a MATLAB function, *sympositivedefinitefactory()*, in software called *Manopt* [21].

mixture model [20, Eq. (2)] , but it is a special case of our formula where $\widehat{\boldsymbol{\Theta}}_i = \theta_i \boldsymbol{I}_d$ with a single parameter θ_i, while $\widehat{\boldsymbol{\Theta}}_i$ in our method is also a diagonal matrix yet the diagonal elements can differ.

It is important to confirm whether the formula (11) updates data samples toward the modes. To this end, we show a sufficient condition that the update formula (11) performs gradient ascent. By multiplying $\left\{\sum_{i=1}^{b} \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i(\boldsymbol{x})\right\}^{-1}$ to the both-hand sides of (10), (11) can be equivalently expressed as

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \left\{\sum_{i=1}^{b} \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i(\boldsymbol{x})\right\}^{-1} \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}). \tag{12}$$

From the definition of ascent direction [22, Sect. 9.2] , (12) shows that the update formula (11) performs gradient ascent whenever

$$\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x})^{\top} \left\{\sum_{i=1}^{b} \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i^{(j)}(\boldsymbol{x})\right\}^{-1} \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}) > 0. \tag{13}$$

Thus, we need to check (13) or that $\left\{\sum_{i=1}^{b} \widehat{\boldsymbol{\Theta}}_i \widehat{\boldsymbol{\Lambda}}_i \widehat{\phi}_i^{(j)}(\boldsymbol{x})\right\}$ is a positive definite matrix when the update formula (11) is used.

Based on this fact, our clustering algorithm consists the following steps:

Step 1 Perform GM-LSLDG and obtain the optimal parameters $\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^{d}$, $\{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^{b}$ and $\{\widehat{\boldsymbol{\Lambda}}_i\}_{i=1}^{b}$.

Step 2 Repeat the following mode-seeking step to each data sample $\boldsymbol{x}_i$ until convergence: If (13) is satisfied, $\boldsymbol{x}_i$ is updated by the fixed-point method (11). Otherwise, we perform standard gradient ascent as $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \epsilon \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{x}_i)$ where ϵ is a fixed step-size parameter.

Step 3 Assign the same cluster label to a set of the data points from which the converged points are close enough.

We empirically observed that most of data samples were monotonically converged to the modes without using standard gradient ascent. Thus, we conjecture that as in MS [6, Theorem 1] and LSLDG clustering [18, Theorem 3] , the update rule (11) has a monotonic hill-climbing property to the modes. Proving this conjecture is our future work.

Finally, we discuss whether our gradient estimator enables us to accurately find the modes of the true density, not of the estimated one. Following the analysis in [18], [23], Appendix A discusses that our gradient estimator with a sufficiently large number of Gaussians would suffice to accurately capture the modes for a wide-range of the density functions as the sample size goes to infinity. However, our experimental results indicate that our clustering method with a small number of Gaussians often works well. Filling this gap between theory and practice is also an interesting challenge, and one of our future works.

4.2 Extension to Hierarchical Clustering

First, we briefly review an existing hierarchical clustering method based on mode-seeking, and then propose our hierarchical clustering method, which is a novel extension of LSLDG clustering.

4.2.1 Hierarchical Mode Association Clustering

A hierarchical clustering method called the *hierarchical mode association clustering* (HMAC) has been proposed based on a mode-seeking clustering method [15]. When kernel density estimation in (4) is employed for HMAC, the main idea is to perform mode-seeking clustering over a set of bandwidth parameters h: A smaller (or larger) value of the bandwidth parameter h produces a less (or more) smooth density estimate, which tends to have more (or less) modes (i.e., clusters). Thus, roughly speaking, a hierarchy of clusters can be built by organizing the clustering results from the set of bandwidth parameters. As discussed in [15], HMAC is substantially different from linkage clustering [24, Sect. 14.3.12] : Clusters in HMAC are merged globally based on the estimated density, while linkage clustering merges two clusters based on local distance, which may result in skewed clusters.

We follow the approach in HMAC, but the extension is not straightforward because we need to prepare for a set of precision matrices in the Gaussian mixture model to obtain a variety of cluster presentations, which seems much more challenging than selecting a set of the bandwidth parameters. Below, we describe how to select a set of the precision matrices and then propose a practical algorithm of our hierarchical clustering method.

4.2.2 Selecting a Set of Precision Matrices

To systematically select a set of precision matrices, we optimize $\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}$ twice from different initializations and store $\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}$ at each iteration to build a hierarchy of clusters. The initialization of $\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}$ is an important problem, and we initialize $\boldsymbol{\Lambda}_i = \eta \boldsymbol{I}_d$ for all i where η is some parameter. Intuitively, when η is a small (or large) value, a set of precision matrices obtained over iterations in the optimization would produce a large (or small) to smaller (or larger) number of clusters. Due to this reason, we optimize $\boldsymbol{\Lambda}_i$ twice from different initializations with a small and large value of η, and unify a set of $\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}$ obtained at each iteration.

However, the difficulty in this approach is that the optimization problem (7) is nonconvex. Thus, the final solutions from two different initializations may not converge to the same solution. To cope with this problem, we first perform GM-LSLDG and obtain $\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^{d}$, $\{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^{b}$ and $\{\widehat{\boldsymbol{\Lambda}}_i\}_{i=1}^{b}$ in (7). Then, given these parameters, we solve the following optimization problem with an additional term:

$$\min_{\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}} \sum_{j=1}^{d} \widehat{J}_j(\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}) + \beta \sum_{i=1}^{b} \|\boldsymbol{\Lambda}_i - \widehat{\boldsymbol{\Lambda}}_i\|_{\mathrm{F}}^2, \tag{14}$$

where $\widehat{J}_j(\{\boldsymbol{\Lambda}_i\}_{i=1}^{b}) := \widehat{J}_j(\widehat{\boldsymbol{\theta}}^{(j)}, \{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^{b}, \{\boldsymbol{\Lambda}_i\}_{i=1}^{b})$, β is a nonnegative parameter, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

The second term in (14) ensures that the final solution is close to $\widehat{\boldsymbol{\Lambda}}_i$ by appropriately choosing β. Thus, we use a set of precision matrices obtained by solving (14) for hierarchical clustering. We choose β as follows: We begin with a small value of β and compute the Frobenius distance between the final solution and $\widehat{\boldsymbol{\Lambda}}_i$ as in the second term of (14). If the Frobenius norm is sufficiently small, then we choose a sequence of precision matrices until the final solution and use it for hierarchical clustering. Otherwise, we slightly increase the value of β, and repeat the same procedure until the Frobenius distance gets sufficiently small.

4.2.3 Practical Algorithm

Before going to the details of the practical algorithm, let us define some notations. We denote by $\mathcal{M}_l$ a set of the mode points obtained at the hierarchical level l. A label partition function of $\boldsymbol{x}$ at the hierarchical level l assigns a cluster label to $\boldsymbol{x}$ and is defined by $P_l(\boldsymbol{x}) \in \{1, 2, \ldots, K_l\}$ where K_l denotes the number of cluster at the level l.

Our algorithm takes the following steps:

Step 1 Perform GM-LSLDG and obtain the optimal parameters $\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^d$, $\{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^b$ and $\{\widehat{\boldsymbol{\Lambda}}_i\}_{i=1}^b$.

Step 2 Given $\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^d$, $\{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^b$, solve (14) with a small value of η (e.g., $\eta = 0.001$) for the initialization $\boldsymbol{\Lambda}_i = \eta \boldsymbol{I}_d$, and obtain a sequence of precision matrices as $\{\tilde{\boldsymbol{\Lambda}}_i^{(1)}, \tilde{\boldsymbol{\Lambda}}_i^{(2)}, \ldots, \tilde{\boldsymbol{\Lambda}}_i^{(T)}\}_{i=1}^b$ where $\tilde{\boldsymbol{\Lambda}}_i^{(t)}$ denotes the precision matrix at the t-th iteration and T is the total number of iterations[†].

Step 3 With a large value of η (e.g., $\eta = 1000$), solve (14) and obtain a sequence of precision matrices as $\{\check{\boldsymbol{\Lambda}}_i^{(1)}, \check{\boldsymbol{\Lambda}}_i^{(2)}, \ldots, \check{\boldsymbol{\Lambda}}_i^{(T)}\}_{i=1}^b$.

Step 4 We unify the two sequences of all precision matrices as $\{\tilde{\boldsymbol{\Lambda}}_i^{(1)}, \ldots, \tilde{\boldsymbol{\Lambda}}_i^{(T)}, \widehat{\boldsymbol{\Lambda}}_i, \check{\boldsymbol{\Lambda}}_i^{(T)}, \ldots, \check{\boldsymbol{\Lambda}}_i^{(1)}\}_{i=1}^b$, and re-express it by $\{\tilde{\boldsymbol{\Lambda}}_i^{(1)}, \ldots, \tilde{\boldsymbol{\Lambda}}_i^{(L)}\}_{i=1}^b$ where $L = 2T + 1$ and $\tilde{\boldsymbol{\Lambda}}_i^{(L)} = \check{\boldsymbol{\Lambda}}_i^{(1)}$. Then, we repeatedly perform GM-LSLDGC with $\{\widehat{\boldsymbol{\theta}}^{(j)}\}_{j=1}^d$, $\{\widehat{\boldsymbol{\mu}}_i\}_{i=1}^b$ and $\{\tilde{\boldsymbol{\Lambda}}_i^{(l)}\}_{i=1}^b$ from $l = 1$ to $l = L$ as follows:

- (a) Perform GM-LSLDGC with $\{\tilde{\boldsymbol{\Lambda}}_i^{(l)}\}_{i=1}^b$ to the points in $\mathcal{M}_{l-1}$ and obtain $\mathcal{M}_l$ where M_0 corresponds to the set of data samples.
- (b) If $P_{l-1}(\boldsymbol{x}_i) = k$ and the k-th mode point in $\mathcal{M}_{l-1}$ converged to the k'-th element in $\mathcal{M}_l$ after mode-seeking, then $P_l(\boldsymbol{x}_i) = k'$.

Step 4 essentially follows HMAC [15] and ensures that this hierarchical clustering algorithm produces a nested clustering representation, i.e., the following mapping holds: $\boldsymbol{x}_i \to M_1(\boldsymbol{x}_i) \to M_2(M_1(\boldsymbol{x}_i)) \to \ldots$ where $M_l(\boldsymbol{x})$ maps $\boldsymbol{x}$ to a mode point in $\mathcal{M}_l$.

Compared with HMAC, the proposed method has an advantage in terms of selection of precision matrices: A set of bandwidth parameters in HMAC is manually selected, while a set of precision matrices in our method is obtained through the optimization, and the parameter manually chosen is only η. Therefore, it should be easier to select a set of precision matrices in our method. In addition, thanks to the precision matrices, our hierarchical clustering method would be more useful than HMAC when the data density includes highly correlated local structures.

[†]To promote the interpretability of the results, when T is large, we only use precision matrices at the $0.7T$-th, $0.8T$-th and $0.9T$-th iterations.

5. Experiments

This section performs numerical experiments both for mode-seeking clustering and hierarchical clustering to demonstrate how the proposed methods work and to compare them with existing methods.

5.1 Mode-Seeking Clustering

Here, we illustrate the performance of the proposed method for mode-seeking clustering both on artificial and benchmark datasets.

5.1.1 Artificial Data

(1) Clustering performance and computational efficiency

We first generated artificial data from the following mixture of three Gaussians:

$$p_{\text{data}}(\boldsymbol{x}) = \sum_{k=1}^{3} \alpha_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k),$$

where $N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k)$ denotes the Gaussian density with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{C}_k$. The mixing coefficients were $\alpha_1 = 0.3$, $\alpha_2 = 0.4$, and $\alpha_3 = 0.3$, respectively. After generating data samples, we applied the following three methods:

- **GM-LSLDGC**: Proposed mode-seeking clustering method based on GM-LSLDG. We performed the ten-fold cross-validation for choosing b from the candidates $\{2, 3, \ldots, 8, 9\}$ with respect to $\sum_{j=1}^d \widehat{J}_j$. We initialized $\boldsymbol{\Lambda}_i$ for all i as a diagonal matrix whose diagonals were sampled from the uniform distribution on $[0.1, 1.0]$. For $\boldsymbol{\mu}_i$, the j-th element was initialized by the uniform distribution on $[\min_i x_i^{(j)}, \max_i x_i^{(j)}]$.
- **LSLDGC** [18]: Mode-seeking clustering based on LSLDG[††]. All the hyper-parameters were determined by the five-hold cross validation.
- **MS** [4]–[6]: Mean shift clustering method. We employed kernel density estimation with a bandwidth matrix $\boldsymbol{H}$ as

$$p_{\text{H}}(\boldsymbol{x}) := \frac{1}{nZ} \sum_{i=1}^{n} \exp\left\{\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^\top \boldsymbol{H}^{-1}(\boldsymbol{x} - \boldsymbol{x}_i)\right\}$$

[††]Software is available at https://sites.google.com/site/hworksites/home/software/lsldg.

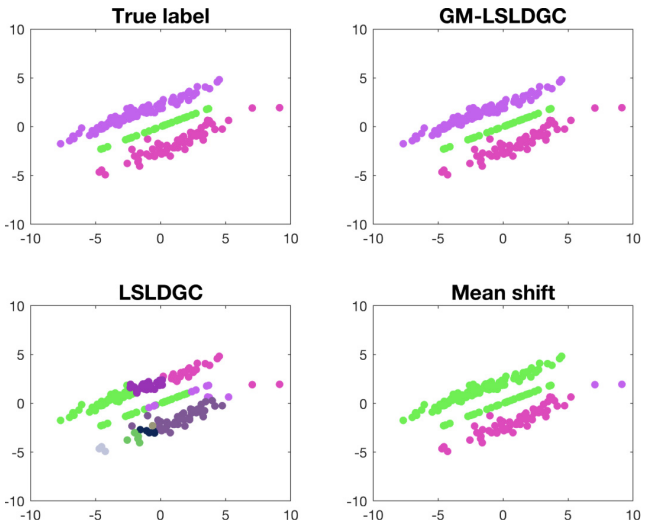


Fig. 1 Examples of mode-seeking clustering results.

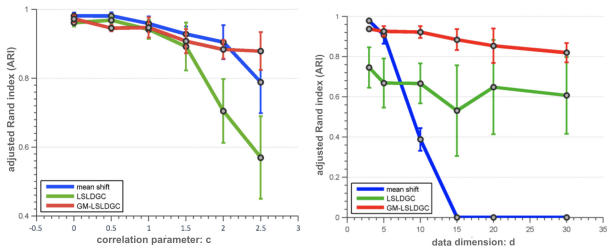


Fig. 2 Clustering performance on artificial data against (left) correlation and (right) data dimension. The markers and error bars denote the means and standard deviations over 30 runs, respectively.

where $Z = (2\pi)^{d/2}|\boldsymbol{H}|^{1/2}$. Based on the well-known normal reference rule [25], $\boldsymbol{H}$ was fixed at $4\widehat{\boldsymbol{C}}n^{-2/(d+6)}/(d+4)$ with the sample covariance matrix $\widehat{\boldsymbol{C}}$, which is optimal in terms of the asymptotic mean integrated squared error for the density derivatives [26, Eq. (4) in $r = 1$] .

We measured the clustering performance by adjusted Rand index (ARI) [27]: ARI takes a value less than or equal to one, a larger value indicates a better clustering result, and when a clustering result is perfect, the ARI value equals to one.

First, to investigate how GM-LSLDGC works to densities with highly correlated structures, we set $\boldsymbol{\mu}_k$ and $\boldsymbol{C}_k$ in p_{data} as follows: $\boldsymbol{\mu}_1 = (1.5, -3.5)^\top$, $\boldsymbol{\mu}_2 = (0.0, 0.0)^\top$ and $\boldsymbol{\mu}_3 = (-1.5, 3.5)^\top$, and

$$\boldsymbol{C}_1 = \begin{pmatrix} 6 & c \\ c & 1.8 \end{pmatrix}, \boldsymbol{C}_2 = \begin{pmatrix} 6 & -c \\ -c & 1.5 \end{pmatrix} \text{ and } \boldsymbol{C}_3 = \begin{pmatrix} 6 & c \\ c & 1.6 \end{pmatrix},$$

where c is a non-negative parameter related to the strength for correlations. Here, we generated $n = 300$ samples. Figure 1 illustrates that GM-LSLDGC well-captures the clusters with highly correlated structures. On the other hand, LSLDGC produces many clusters, which implies that it is not easy to obtain a smooth gradient estimate by the spherical Gaussian kernel. The left plot in Fig. 2 quantitatively shows that GM-LSLDGC and MS perform well to the highly correlated structures because these methods employ non-spherical Gaussian functions, while LSLDGC gets worse as the correlation parameter c becomes larger.

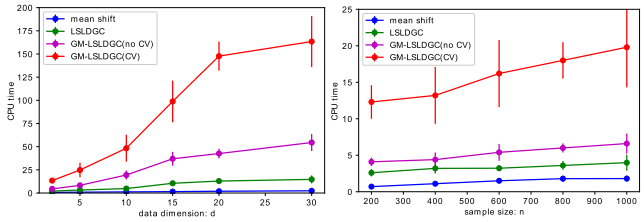


Fig. 3 Comparison in term of CPU time in seconds against (left) data dimension in $n = 200$ and (right) sample size in $d = 2$. The markers and error bars denote the means and standard deviations over 30 runs, respectively. GM-LSLDGC (no CV) denotes GM-LSLDGC without cross-validation where the number of Gaussians is fixed at 4.

Next, we demonstrate the performance of GM-LSLDGC to higher-dimensional data. Here, we set $\boldsymbol{C}_k$ in p_{data} as follows: the diagonals in all $\boldsymbol{C}_k$ are all one and the $(1,2)$-th and $(2,1)$-th in $\boldsymbol{C}_1$, $(1,3)$-th and $(3,1)$-th in $\boldsymbol{C}_2$ and $(2,3)$-th and $(3,2)$-th elements in $\boldsymbol{C}_3$ are 0.75, while the other elements in all $\boldsymbol{C}_k$ are zeros. Regarding $\boldsymbol{\mu}_k$, we appended zeros to the mean parameters in the previous experiment. $n = 200$ samples were generated. The right plot in Fig. 2 shows the superior performance of GM-LSLDGC to higher-dimensional data. LSLDGC fairly works, but the performance of MS rapidly decreases as previously demonstrated in [17], [18]. A possible reason is that combined with the direct fitting, our model (6) is more adaptive by precision matrices $\boldsymbol{\Lambda}_i$ as well as coefficient parameters $\theta_i^{(j)}$.

Finally, we investigate the computational efficiency of GM-LSLDGC. To this end, we performed the same experiment as the right plot in Fig. 2. Figure 3 shows that the computationally most efficient method is mean shift and LSLDGC is also fairly efficient. On the other hand, GM-LSLDGC is the most computationally demanding method. This is due to the fact that unlike mean shift and LSLDGC, an iterative optimization procedure is adopted to estimate the Gaussian mixture model together with cross-validation for the number of Gaussians in the mixture model. The computational cost of GM-LSLDGC can be alleviated without performing cross-validation (Fig. 3). This approach is promising because as will be shown in Fig. 4, the good clustering performance is retained as long as we use an enough number of Gaussians.

In short, GM-LSLDGC can be advantageous when data is high-dimensional and includes highly correlated local structures, while the superior clustering performance comes at the price of increasing the computational cost. As a remedy, the computational cost could be alleviated by fixing the number of Gaussians in advance.

(2) Influence of the number of Gaussians

Here, we show how the number of Gaussians affects the clustering performance of GM-LSLDGC. In this experiment, we fixed the number of Gaussians and performed GM-LSLDGC without cross-validation. The same two-

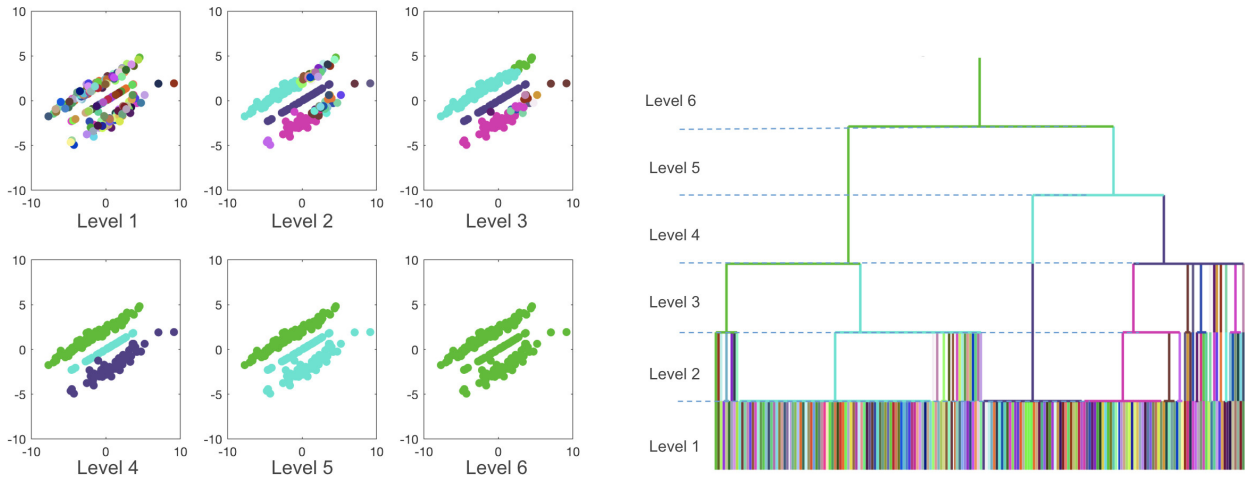


Fig. 5 (Left) A cluster-merging process and (Right) dendrogram. The levels in the dendrogram correspond to the cluster representations of the left plots.

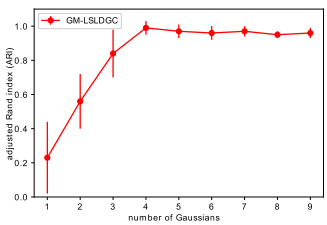


Fig. 4 Clustering performance against the number of Gaussians. The same two-dimensional data as Fig. 1 was used. The markers and error bars denote the means and standard deviations over 30 runs, respectively.

dimensional data as Fig. 1 was used.

Figure 4 indicates that accurate clustering is possible as long as an enough number of Gaussians is used. This would be for the reason that a larger number of Gaussians can potentially approximate a wider-range of functions, and thus a better gradient estimate was obtained with an enough number of Gaussians. This property of GM-LSLDGC is stark contrast with standard mixture-model-based clustering where the number of clusters corresponds to the number of mixtures [7]. Therefore, GM-LSLDGC would be easier to determine the number of Gaussians.

5.1.2 Benchmark Data

Here, we investigate the clustering performance on benchmark datasets available at the UCI machine learning repository[†]. Since we used classification datasets, we regarded the class labels as the true cluster labels. In addition to the three methods in the previous experiments, we applied the k-means clustering method where the number of clusters was fixed at the true cluster number. Before applying the clustering methods, data samples were standardized by subtracting the sample mean and normalizing with the sample variance.

Table 1 shows that GM-LSLDGC performs better than MS and k-means, and compares favorably with LSLDGC on benchmark datasets.

[†]https://archive.ics.uci.edu/ml/index.php

Table 1 The average and standard deviation of ARI values over 20 runs. Numbers in the parentheses are standard deviations. The best and comparable methods judged by the unpaired t-test at the significance level 5% are described in boldface. k denotes the true number of clusters.

GM-LSLDGC	LSLDGC	Mean shift	k-means
Electricity ($d = 2, n = 400, k = 4$)			
0.493(0.083)	0.367(0.035)	0.379(0.014)	0.365(0.055)
Yeast ($d = 8, n = 250, k = 5$)			
0.662(0.029)	**0.651(0.044)**	0.118(0.003)	0.643(0.041)
Anuran Calls ($d = 18, n = 150, k = 3$)			
0.298(0.072)	0.255(0.048)	0.000(0.000)	0.237(0.029)
Sports articles ($d = 59, n = 200, k = 2$)			
0.224(0.068)	**0.323(0.116)**	0.000(0.000)	0.000(0.000)

5.2 Hierarchical Clustering

Finally, we perform the experiments for hierarchical clustering. To first visualize the cluster-merging process in our method, artificial data were generated as in Fig. 1. The left plot in Fig. 5 shows the cluster-merging process. Initially, all the data points are regarded as individual clusters, but as the hierarchy level becomes higher, the clusters are merged. This merging process can be summarized in the dendrogram of Fig. 5.

Next, we compare our hierarchical clustering method with HMAC on artificial and benchmark datasets. As a performance measure for hierarchical clustering, we used FScore [28], which takes a value between zero and one and whose larger value means better hierarchical clustering. For HMAC, we downloaded the software at http://personal.psu.edu/jol2/hmac/ and used it as the default setting.

Table 2 shows FScores on artificial and benchmark datasets downloaded from the UCI machine learning repository. Our hierarchical clustering method gets higher values of FScore than HMAC. This could be due to the fact that our method employs a set of precision matrices, while HMAC uses a set of bandwidth parameters. Thus, our method could

Table 2 The average and standard deviation of FScores over 20 runs. Numbers in the parentheses are standard deviations. The best and comparable methods judged by the unpaired t-test at the significance level 5% are described in boldface. h-GM-LSLDGC indicates the proposed hierarchical clustering method. k denotes the true number of clusters.

h-GM-LSLDGC	HMAC
Artificial Data ($d = 2, n = 350, k = 3$)	
0.889(0.163)	0.651(0.217)
Electricity ($d = 2, n = 400, k = 4$)	
0.467(0.294)	0.230(0.091)
Iris ($d = 4, n = 90, k = 3$)	
0.595(0.174)	0.547(0.036)

better-capture locally correlated structures than HMAC.

6. Conclusion

In this paper, we proposed a new estimator of the gradient of a logarithmic probability density function. In contrast with an existing estimator, we propose to use Gaussian mixture models, which enable the new estimator to flexibly capture the highly correlated structures in the probability density function. We applied the proposed estimator to mode-seeking clustering, and developed a fixed-point algorithm for mode-seeking. Our experimental results showed that our mode-seeking clustering method is advantageous particularly when the dimensionality of data is relatively high and clusters have highly correlated structures, although the good performance comes at a cost of making our clustering method computationally more demanding. A promising remedy is to use an enough number of Gaussians without performing cross-validation, which could decrease the computational cost without sacrificing the clustering performance. Finally, we extended our clustering method to hierarchical clustering, and demonstrated its usefulness.

This paper focused only on mode-seeking clustering and hierarchical clustering as applications of the new gradient estimator. However, the proposed estimator is potentially applicable to other problems such as unsupervised dimensionality reduction [1], modal regression [2] and so on. In the future, we explore new applications of the proposed estimator.

References

[1] H. Sasaki, G. Niu, and M. Sugiyama, "Non-Gaussian component analysis with log-density gradient estimation," Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), pp.1177–1185, 2016.
[2] J. Einbeck and G. Tutz, "Modelling beyond regression functions: an application of multimodal regression to speed–flow data," Journal of the Royal Statistical Society: Series C (Applied Statistics), vol.55, no.4, pp.461–475, 2006.
[3] R. Singh, "Applications of estimators of a density and its derivatives to certain statistical problems," Journal of the Royal Statistical Society. Series B, vol.39, no.3, pp.357–363, 1977.
[4] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," IEEE Transactions on Information Theory, vol.21, no.1, pp.32–40, 1975.
[5] Y. Cheng, "Mean shift, mode seeking, and clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, no.8, pp.790–799, 1995.
[6] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no.5, pp.603–619, 2002.
[7] V. Melnykov and R. Maitra, "Finite mixture models and model-based clustering," Statistics Surveys, vol.4, pp.80–116, 2010.
[8] W. Tao, H. Jin, and Y. Zhang, "Color image segmentation based on mean shift and normalized cuts," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.37, no.5, pp.1382–1389, 2007.
[9] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, "Image and video segmentation by anisotropic kernel mean shift," Proceedings of European Conference on Computer Vision (ECCV), vol.3022, pp.238–249, 2004.
[10] R.T. Collins, "Mean-shift blob tracking through scale space," Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp.234–240, 2003.
[11] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.142–149, 2000.
[12] M. Carreira-Perpiñán, "A review of mean-shift algorithms for clustering," arXiv preprint arXiv:1503.00687, 2015.
[13] R. Subbarao and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," Proceedings of IEEE Computer Society Conference on of Computer Vision and Pattern Recognition (CVPR), pp.1168–1175, 2006.
[14] R. Subbarao and P. Meer, "Nonlinear mean shift over Riemannian manifolds," International Journal of Computer Vision, vol.84, no.1, pp.1–20, 2009.
[15] J. Li, S. Ray, and B.G. Lindsay, "A nonparametric statistical approach to clustering via mode identification," Journal of Machine Learning Research, vol.8, pp.1687–1723, 2007.
[16] D.D. Cox, "A penalty method for nonparametric estimation of the logarithmic derivative of a density function," Annals of the Institute of Statistical Mathematics, vol.37, no.1, pp.271–288, 1985.
[17] H. Sasaki, A. Hyvärinen, and M. Sugiyama, "Clustering via mode seeking by direct estimation of the gradient of a log-density," Machine Learning and Knowledge Discovery in Databases Part III- European Conference, ECML/PKDD 2014, pp.19–34, 2014.
[18] H. Sasaki, T. Kanamori, A. Hyvärinen, G. Niu, and M. Sugiyama, "Mode-seeking clustering and density ridge estimation via direct estimation of density-derivative-ratios," Journal of Machine Learing Research, vol.18, no.180, 2018.
[19] P.-A. Absil, R. Mahony, and R. Sepulchre, "Optimization algorithms on matrix manifolds," Princeton University Press, 2008.
[20] M. Carreira-Perpiñán, "Gaussian mean-shift is an EM algorithm," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.29, no.5, pp.767–776, 2007.
[21] N. Boumal, B. Mishra, P. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," Journal of Machine Learning Research, vol.15, pp.1455–1459, 2014.
[22] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[23] Y.-C. Chen, C.R. Genovese, and L. Wasserman, "A comprehensive approach to mode clustering," Electronic Journal of Statistics, vol.10, no.1, pp.210–241, 2016.
[24] T. Hastie, R. Tibshirani, and J. Friedman, "Ensemble Learning," The elements of statistical learning, pp.605–624, Springer, 2009.
[25] B. Silverman, Density Estimation for Statistics and Data Analysis, CRC Press, 1986.
[26] J.E. Chacón, T. Duong, and M. Wand, "Asymptotics for general multivariate kernel density derivative estimators," Statistica Sinica, vol.21, pp.807–840, 2011.
[27] L. Hubert and P. Arabie, "Comparing partitions," Journal of Classification, vol.2, no.1, pp.193–218, 1985.
[28] B. Larsen and C. Aone, "Fast and effective text mining using linear-

time document clustering," Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp.16–22, ACM, 1999.

[29] H. Sasaki and A. Hyvärinen, "Neural-kernelized conditional density estimation," arXiv preprint arXiv:1806.01754, 2018.

[30] J. Park and I. Sandberg, "Universal approximation using radial-basis-function networks," Neural computation, vol.3, no.2, pp.246–257, 1991.

Appendix A: Accuracy of Mode Estimation

Let us define the mode $\boldsymbol{m}_j$ of the true logarithmic probability density function as a point satisfying

$$\boldsymbol{g}(\boldsymbol{m}_j) = \boldsymbol{0} \text{ and } \nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{m}_j) \preceq \boldsymbol{O}, \tag{A·1}$$

where $\boldsymbol{g}(\boldsymbol{x}) := \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$, $\nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x}) := \nabla_{\boldsymbol{x}}\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$, and $\nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{m}_j) \preceq \boldsymbol{O}$ means that the Hessian matrix $\nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x})$ is negative definite at $\boldsymbol{x} = \boldsymbol{m}_j$. Here, we follow the analysis in [18], [23]. First of all, let us make two assumptions: (1) The matrix $\nabla_{\boldsymbol{x}}\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j)$ is invertible, and (2) each mode $\boldsymbol{m}_j$ is uniquely approximated by an estimated mode $\widehat{\boldsymbol{m}}_j$ which is defined as a point satisfying

$$\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\widehat{\boldsymbol{m}}_j) = \boldsymbol{0} \text{ and } \nabla_{\boldsymbol{x}}\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\widehat{\boldsymbol{m}}_j) \preceq \boldsymbol{O}, \tag{A·2}$$

With the definition (A·2), the Taylor expansion yields

$$\begin{aligned}\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j) &= \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j) - \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\widehat{\boldsymbol{m}}_j) \\ &= \nabla_{\boldsymbol{x}}\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j)(\widehat{\boldsymbol{m}}_j - \boldsymbol{m}_j) + o(\|\widehat{\boldsymbol{m}}_j - \boldsymbol{m}_j\|).\end{aligned} \tag{A·3}$$

On the other hand, by the definition (A·1), we have

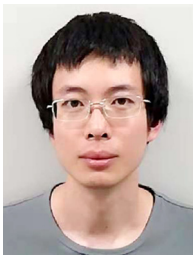
$$\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j) = \widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j) - \boldsymbol{g}(\boldsymbol{m}_j) \tag{A·4}$$

Combining (A·3) with (A·4) yields

$$\|\widehat{\boldsymbol{m}}_j - \boldsymbol{m}_j\| \leq O(\|\widehat{\boldsymbol{g}}^{\mathrm{GM}}(\boldsymbol{m}_j) - \boldsymbol{g}(\boldsymbol{m}_j)\|). \tag{A·5}$$

Thus, a mode $\boldsymbol{m}_j$ can be well-approximated by a mode estimate $\widehat{\boldsymbol{m}}_j$ if $\widehat{\boldsymbol{g}}^{\mathrm{GM}}$ is an accurate estimate of $\boldsymbol{g}$ around $\boldsymbol{m}_j$.

Since we directly fit a model $\boldsymbol{g}^{\mathrm{GM}}$ to $\boldsymbol{g}$ under the squared-loss, $\boldsymbol{g}^{\mathrm{GM}}$ converges to $\boldsymbol{g}$ as the samples size n goes to infinity if $\boldsymbol{g}^{\mathrm{GM}}$ and $\boldsymbol{g}$ belong to the same function set (See Proposition 3 in [29] for a more rigorous analysis based on the same squared loss). Thus, we next need to confirm whether a function set of $\boldsymbol{g}^{\mathrm{GM}}$ is large enough. The Gaussian mixture model in (6) includes radial basis function (RBF) networks with Gaussian functions as a special case, and RBF networks with a sufficiently large number of Gaussians have the universal approximation capability under some conditions [30]. Since $\boldsymbol{g}^{\mathrm{GM}}$ is the gradient of the Gaussian mixture model, $\boldsymbol{g}^{\mathrm{GM}}$ with a sufficiently large number of Gaussians could well-approximate a wide-range of continuous $\boldsymbol{g}$ as the samples size n approaches infinity. To make a more rigorous statement, we need to establish the consistency of $\widehat{\boldsymbol{g}}^{\mathrm{GM}}$ under the uniform norm $\max_j \sup_{\boldsymbol{x}} |\widehat{g}_j^{\mathrm{GM}}(\boldsymbol{x}) - g_j(\boldsymbol{x})|$, but this is beyond the scope of this paper. Therefore, we leave this challenge for our future work.



Zhang Qi received his B.E. in Traffic and Transportation from Henan University of Science and Technology, China and M.E. in Information Science from Nara Institute of Science and Technology, Japan in 2014 and 2018, respectively. He is currently pursuing the doctoral degree from the Graduate University for Advanced Studies, Japan. His research interests include applications and theories of machine learning.

Hiroaki Sasaki received his B.E. in Applied Physics, and M.E. and D.E. in Electrical and Communication Engineering from Tohoku University in 2006, 2008 and 2011, respectively. He was a research fellow of the Japan Society for the Promotion of Science from 2011 to 2014. He was a postdoctoral researcher with the Graduate School of Information Science and Engineering, Tokyo Institute of Technology for six months in 2014, and with the Graduate School of Frontier Sciences, the University of Tokyo from 2014 to 2015. Since 2016, he has been an assistant professor of Nara Institute of Science and Technology. His research interests include algorithms and theories of machine learning.

Kazushi Ikeda received his B.E., M.E., and Ph.D. in mathematical engineering and information physics from the University of Tokyo in 1989, 1991, and 1994. He was a research associate with the Department of Electrical and Computer Engineering of Kanazawa University from 1994 to 1998. He was a research associate of Chinese University of Hong Kong for three months in 1995. He was with Graduate School of Informatics, Kyoto University, as an associate professor from 1998 to 2008. Since 2008, he has been a full professor of Nara Institute of Science and Technology. He was the editor-in-chief of the Journal of the Japanese Neural Network Society, and is currently an action editor of Neural Networks, and an associate editor of IEEE Transactions on Neural Networks and Learning Systems.