Clustering via Mode Seeking by Direct Estimation of the Gradient of a Log-Density

Hiroaki Sasaki[1], Aapo Hyvärinen[2,3], and Masashi Sugiyama[1]

1 Graduate School of Information Science and Engineering,
Tokyo Institute of Technology, Tokyo, Japan
2 Department of Computer Science and HIIT,
University of Helsinki, Helsinki, Finland
3 Cognitive Mechanisms Laboratories, ATR, Kyoto, Japan
{sasaki@sg.,sugi@}cs.titech.ac.jp aapo.hyvarinen@helsinki.fi

Abstract. *Mean shift clustering* finds the *modes* of the data probability density by identifying the zero points of the density gradient. Since it does not require to fix the number of clusters in advance, the mean shift has been a popular clustering algorithm in various application fields. A typical implementation of the mean shift is to first estimate the density by kernel density estimation and then compute its gradient. However, since a good density estimation does not necessarily imply an accurate estimation of the density gradient, such an indirect two-step approach is not reliable. In this paper, we propose a method to *directly* estimate the gradient of the log-density without going through density estimation. The proposed method gives the global solution analytically and thus is computationally efficient. We then develop a mean-shift-like fixed-point algorithm to find the modes of the density for clustering. As in the mean shift, one does not need to set the number of clusters in advance. We experimentally show that the proposed clustering method significantly outperforms the mean shift especially for high-dimensional data.

Keywords: Log-Density Gradient Estimation, Mean Shift, Clustering, High-Dimensional Data

1 Introduction

Seeking the *modes* of a probability density has led to a powerful clustering algorithm called the *mean shift* [6, 8, 11]. In the mean shift algorithm, all input samples are initially regarded as candidates of the modes of the density and they are iteratively updated and merged. Finally, clustering is performed by associating the input samples with the obtained modes. An advantage of the mean shift is that the number of clusters does not need to be specified in advance. Thanks to this extremely useful property, the mean shift has been successfully employed in various applications such as image segmentation [8, 24, 26] and object tracking [7, 9].

In mode seeking, a central technical challenge is accurate estimation of the gradient of a density. The mean shift takes a two-step approach: kernel density

estimation (KDE) is first used to approximate the density and then its gradient is computed. However, such a two-step approach performs poorly because a good estimator of the density does not necessarily mean a good estimator of the density gradient. In particular, KDE tends to produce a smooth density estimate and therefore the modes in a multi-modal density could be collapsed. Furthermore, KDE itself tends to perform poorly in high-dimensional problems [8].

To overcome this problem, we propose a method called the *least-squares log-density gradient* (LSLDG), which *directly* estimates the gradient of a log-density by least-squares without going through density estimation. The proposed method can be regarded as a non-parametric extension of *score matching* [14, 21], which has originally been developed for least-squares parametric density estimation with intractable partition functions. We then derive a fixed-point algorithm to find the modes of the density, which is our proposed clustering algorithm called *LSLDG clustering*.

All tuning parameters included in LSLDG such as the Gaussian kernel width and the regularization parameter can be objectively optimized by cross-validation in terms of the squared error. Furthermore, since LSLDG clustering inherits the same algorithmic structure as the original mean shift, it does not require the number of clusters to be fixed in advance. Thus, LSLDG clustering does not involve *any* tuning parameters to be manually determined, which is a significant advantage over standard clustering algorithms such as *spectral clustering* [19], because clustering is an unsupervised learning problem and appropriately controlling tuning parameters is generally very hard. A recent study based on *information-maximization clustering* [22] provided an information-theoretic mean to determine tuning parameters objectively, but it still requires the user to fix the number of clusters in advance.

The remainder of this paper is structured as follows. We derive a method to directly estimate the gradient of a log-density in Section 2, and then use it for finding clusters in the data in Section 3. Various possibilities for extension are discussed in Section 4. The usefulness of the proposed method is experimentally investigated in Section 5. Finally this paper is concluded in Section 6.

2 Direct Estimation of the Gradient of a Log-density

In this section, we propose a method to estimate the log-density gradient.

2.1 Problem Formulation

Let us consider a probability distribution on $\mathbb{R}^d$ with density $p^*(\boldsymbol{x})$, which is unknown but n i.i.d. samples $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ are available:

$$\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \overset{\text{i.i.d}}{\sim} p^*(\boldsymbol{x}).$$

Our goal is to estimate the gradient of the logarithm of the density $p^*(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ from $\mathcal{X}$:

$$\boldsymbol{g}^*(\boldsymbol{x}) = (g_1^*(\boldsymbol{x}), \ldots, g_d^*(\boldsymbol{x}))^\top = \nabla \log p^*(\boldsymbol{x}) = \frac{\nabla p^*(\boldsymbol{x})}{p^*(\boldsymbol{x})}.$$

A naive approach to estimate $\boldsymbol{g}^*(\boldsymbol{x})$ is to first obtain a density estimate $\widehat{p}(\boldsymbol{x})$ and then compute its log-gradient $\nabla \log \widehat{p}(\boldsymbol{x})$. However, this two-step approach does not work well because a good density estimate $\widehat{p}(\boldsymbol{x})$ does not necessarily provide an accurate estimate of its log-gradient $\nabla \log \widehat{p}(\boldsymbol{x})$. For example, in Figures 1(a) and (b), density estimation is performed very well by KDE, but its log-density gradient produces oscillated errors. These errors become more prominent especially in higher-dimensional data (Figure 1(c)).

Below, we describe a method to directly estimate the log-density gradient $\nabla \log p^*(\boldsymbol{x})$ without going through density estimation. Our proposed method is based on the mathematics of score matching [14]; the difference is that our goal is to approximate the gradient of the log-density instead of model parameter estimation.

2.2 Least-Squares Log-Density Gradient

Our basic idea is to directly fit a model $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_d(\boldsymbol{x}))^\top$ to the true log-density gradient $\boldsymbol{g}^*(\boldsymbol{x})$ under the squared loss:

$$\begin{aligned}
J_j(g_j) &= \int \Big(g_j(\boldsymbol{x}) - g_j^*(\boldsymbol{x})\Big)^2 p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \int g_j^*(\boldsymbol{x})^2 p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\
&= \int g_j(\boldsymbol{x})^2 p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - 2 \int g_j(\boldsymbol{x}) g_j^*(\boldsymbol{x}) p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\
&= \int g_j(\boldsymbol{x})^2 p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - 2 \int g_j(\boldsymbol{x}) \partial_j p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\
&= \int g_j(\boldsymbol{x})^2 p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + 2 \int \partial_j g_j(\boldsymbol{x}) p^*(\boldsymbol{x}) \mathrm{d}\boldsymbol{x},
\end{aligned}$$

where ∂_j denotes the partial derivative with respect to the j-th variable of $\boldsymbol{x}$ and the last equality follows from *integration by parts* under some conditions [14]. Then the empirical approximation of J_j is given as

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

As the model $g_j(\boldsymbol{x})$, we use the following linear-in-parameter model, which is related to using an exponential family for density modeling:

$$g_j(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_{i,j} \psi_{i,j}(\boldsymbol{x}) = \boldsymbol{\theta}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}),$$

where $\boldsymbol{\theta}_j$ denotes the parameter vector and $\psi_{i,j}(\boldsymbol{x})$ is a basis function. The derivative of this model is given by

$$\partial_j g_j(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_{i,j} \partial_j \psi_{i,j}(\boldsymbol{x}) = \boldsymbol{\theta}_j^\top \boldsymbol{\varphi}_j(\boldsymbol{x}),$$

where $\boldsymbol{\varphi}_j(\boldsymbol{x}) = (\partial_j \psi_{1,j}(\boldsymbol{x}), \ldots, \partial_j \psi_{n,j}(\boldsymbol{x}))$.

Adding an ℓ_2-regularizer to (1), we can compactly express the optimization problem as

$$\widehat{\boldsymbol{\theta}}_j = \arg\min_{\boldsymbol{\theta}_j} \left[\boldsymbol{\theta}_j^\top \boldsymbol{G}^{(j)} \boldsymbol{\theta}_j + 2\boldsymbol{\theta}_j^\top \boldsymbol{h}_j + \lambda \boldsymbol{\theta}_j^\top \boldsymbol{\theta}_j \right], \tag{2}$$

where $\lambda \geq 0$ is the regularization parameter, and $\boldsymbol{G}^{(j)}$ and $\boldsymbol{h}_j$ are defined by

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

As in score matching for an exponential family [15], the optimization problem (2) can be solved analytically as

$$\widehat{\boldsymbol{\theta}}_j = -(\boldsymbol{G}^{(j)} + \lambda \boldsymbol{I})^{-1} \boldsymbol{h}_j,$$

where $\boldsymbol{I}$ denotes the identity matrix. Finally, we obtain the estimator $\widehat{g}_j$ as

$$\widehat{g}_j(\boldsymbol{x}) = \sum_{i=1}^{n} \widehat{\theta}_{i,j} \psi_{i,j}(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}).$$

We call this method the *least-squares log-density gradient* (LSLDG).

2.3 Model Selection by Cross-Validation

The performance of LSLDG depends on the choice of the regularization parameter λ and parameters included in the basis function $\boldsymbol{\psi}_j$. They can be objectively chosen via cross-validation as follows:

1. Divide the samples $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ into N disjoint subsets $\{\mathcal{X}_i\}_{i=1}^N$.
2. For $i = 1, \ldots, N$
   (a) Compute the LSLDG estimator $\widehat{g}_j^{(i)}$ from $\mathcal{X} \backslash \mathcal{X}_i$ (i.e., all samples except $\mathcal{X}_i$).
   (b) Compute its hold-out error for $\mathcal{X}_i$:

$$\mathrm{CV}^{(i)} = \frac{1}{|\mathcal{X}_i|} \sum_{\boldsymbol{x} \in \mathcal{X}_i} \sum_{j=1}^{d} \left[\widehat{g}_j^{(i)}(\boldsymbol{x})^2 + 2\partial_j \widehat{g}_j^{(i)}(\boldsymbol{x}) \right],$$

where $|\mathcal{X}_i|$ denotes the cardinality of $\mathcal{X}_i$.

3. Compute the average hold-out error as

$$\mathrm{CV} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{CV}^{(i)}. \tag{3}$$

4. Choose the model that minimizes (3) with respect to λ and parameters in $\boldsymbol{\psi}_j$, and compute the final LSLDG estimator $\widehat{g}_j$ with the chosen model using all samples $\mathcal{X}$.

3 Clustering via Mode Seeking

In this section, we derive a clustering algorithm based on LSLDG. Our basic idea follows the same line as the *mean shift* algorithm [6, 8, 11], i.e., to assign each data sample to a nearby *mode* of the density.

3.1 Gradient-Based Approaches

A naive implementation of this idea is to use *gradient ascent* for each data sample to let it converge to one of the modes of the density in the vicinity:

$$\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i + \varepsilon \widehat{\boldsymbol{g}}(\boldsymbol{x}_i),$$

where $\varepsilon > 0$ is the step size.

Since

$$\boldsymbol{g}(\boldsymbol{x}) = \nabla \log p(\boldsymbol{x}) = \frac{\nabla p(\boldsymbol{x})}{p(\boldsymbol{x})} \propto \nabla p(\boldsymbol{x}),$$

the gradient of the log-density $\log p(\boldsymbol{x})$ keeps the same direction as the gradient of the original density $p(\boldsymbol{x})$. However, due to $p(\boldsymbol{x})$ in the denominator, the log-gradient vector gets longer when $p(\boldsymbol{x}) < 1$ and shorter when $p(\boldsymbol{x}) > 1$. This is practically suitable adjustment because $p(\boldsymbol{x}) < 1$ ($p(\boldsymbol{x}) > 1$) often means that the current point $\boldsymbol{x}$ is far from (close to) a mode. Indeed, the faster convergence of gradient ascent with the log-density was asserted in the same way [11].

To further increase the speed of convergence, using a *quasi-Newton* method is also promising:

$$\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i + \varepsilon \widehat{\boldsymbol{Q}}\widehat{\boldsymbol{g}}(\boldsymbol{x}_i),$$

where $\widehat{\boldsymbol{Q}}$ is an estimate of the inverse Hessian matrix.

3.2 Fixed-Point Approach

In the gradient-based approaches, choosing the step size parameter ε is a crucial problem. To avoid this problem, we develop a *fixed-point method*, in analogy to the original mean-shift method.

To easily derive a fixed-point equation, we focus on the basis function of the following form:

$$\psi_{i,j}(\boldsymbol{x}) = \frac{1}{\sigma^2}[\boldsymbol{c}_i - \boldsymbol{x}]_j \phi_i(\boldsymbol{x}),$$

where σ^2 is a constant, $\boldsymbol{c}_i$ is a d-dimensional constant vector, $\phi_i(\boldsymbol{x})$ is a "mother" basis function, and $[\cdot]_j$ denotes the j-th element of a vector. A typical choice of the mother basis function $\phi_i(\boldsymbol{x})$ is the Gaussian function:

$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma^2}\right), \tag{4}$$

where the Gaussian center $\boldsymbol{c}_i$ may be fixed at sample $\boldsymbol{x}_i$. In experiments, we only use 100 Gaussian centers chosen randomly from $\mathcal{X}$. This reduction of Gaussian centers significantly decreases the computational costs without sacrificing the performance, as shown in Section 5.2.

For this model, the LSLDG solution can be expressed as

$$\begin{aligned}\widehat{g}_j(\boldsymbol{x}) &= \sum_{i=1}^n \widehat{\theta}_{i,j}\psi_{i,j}(\boldsymbol{x}) = \frac{1}{\sigma^2}\sum_{i=1}^n \widehat{\theta}_{i,j}[\boldsymbol{c}_i - \boldsymbol{x}]_j \phi_i(\boldsymbol{x}) \\ &= \frac{1}{\sigma^2}\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})[\boldsymbol{c}_i]_j - \frac{[\boldsymbol{x}]_j}{\sigma^2}\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x}).\end{aligned}$$

If $\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x}) \neq 0$, setting $\widehat{g}_j(\boldsymbol{x})$ to zero yields

$$[\boldsymbol{x}]_j = \frac{\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})[\boldsymbol{c}_i]_j}{\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})}. \tag{5}$$

We propose to use this equation as a fixed-point update formula by iteratively substituting the right-hand side to the left-hand side. In the vector-matrix form, the update formula is compactly expressed as

$$\boldsymbol{x}_i \longleftarrow \boldsymbol{B}\boldsymbol{\phi}(\boldsymbol{x}_i)./(\widehat{\boldsymbol{\Theta}}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)),$$

where $B_{j,i} = \widehat{\theta}_{i,j}[\boldsymbol{c}_i]_j$, $\widehat{\Theta}_{i,j} = \widehat{\theta}_{i,j}$, $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_n(\boldsymbol{x}))^\top$, and "./" denotes the element-wise division.

This update formula is similar to the one used in the original mean shift algorithm [8, Eq.(20)], which corresponds to $\widehat{\theta}_{i,j} = 1/n$:

$$\boldsymbol{x} \longleftarrow \frac{\sum_{i=1}^n \phi_i(\boldsymbol{x})\boldsymbol{c}_i}{\sum_{i=1}^n \phi_i(\boldsymbol{x})},$$

where ϕ_i is typically chosen as the Gaussian function (4). Thus, the proposed method can be regarded as a weighted variant of the mean shift algorithm, where the weights $\widehat{\theta}_{i,j}$ are learned by LSLDG. A similar weighted mean shift method has already been studied in [6], but the weights were determined heuristically.

The mean shift update was proven to be equivalent to gradient ascent with an adaptive step size [6]. LSLDG-based clustering also inherits this property. Indeed, if $[\boldsymbol{x}]_j \sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})$ is subtracted from and added to the numerator of Eq.(5) (thus the equation remains the same), we obtain

$$[\boldsymbol{x}]_j = [\boldsymbol{x}]_j + \varepsilon_j(\boldsymbol{x})\widehat{g}_j(\boldsymbol{x}),$$

where

$$\varepsilon_j(\boldsymbol{x}) = \frac{\sigma^2}{\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})}.$$

This shows that our fixed-point update rule can be regarded as gradient ascent with an adaptive step size $\varepsilon_j(\boldsymbol{x})$.

If $\phi_i(\boldsymbol{x})$ is set to be the Gaussian function (4), $\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})$ can actually be regarded as an estimate of the original log-density $\log p^*(\boldsymbol{x})$. More specifically, we can easily see that the partial derivative of $\phi_i(\boldsymbol{x})$ with respect to the j-th variable of $\boldsymbol{x}$ is $\psi_{i,j}(\boldsymbol{x})$:

$$\partial_j \phi_i(\boldsymbol{x}) = \psi_{i,j}(\boldsymbol{x}).$$

Then we have

$$\begin{aligned}\partial_j \log p^*(\boldsymbol{x}) = g_j^*(\boldsymbol{x}) \approx \widehat{g}_j(\boldsymbol{x}) &= \sum_{i=1}^n \widehat{\theta}_{i,j}\psi_{i,j}(\boldsymbol{x}) \\ &= \sum_{i=1}^n \widehat{\theta}_{i,j}\partial_j\phi_i(\boldsymbol{x}) = \partial_j \sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x}).\end{aligned}$$

This implies that $\sum_{i=1}^n \widehat{\theta}_{i,j}\phi_i(\boldsymbol{x})$ is an estimate of $\log p^*(\boldsymbol{x})$ up to a constant. Therefore, when $\log p^*(\boldsymbol{x})$ is small (large), the proposed fixed-point algorithm adaptively increases (decreases) the step size $\varepsilon_j(\boldsymbol{x})$ to more aggressively (conservatively) ascend the gradient. This step-size adaptation would be reasonable because small (large) $\log p^*(\boldsymbol{x})$ often means that the current solution is far from (close to) a mode.

4 Extensions

In the previous section, we focused on the simplest setting to clearly convey the essence of the proposed idea. However, we can easily extend the proposed method to various directions. In this section, we discuss such possibilities.

4.1 Common Basis Functions

When the basis function is common to all dimensions, i.e., $\boldsymbol{\psi}_j(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})$ for $j = 1, \ldots, d$, the matrix $\boldsymbol{G}^{(j)}$ becomes independent of j as $\boldsymbol{G} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{\psi}(\boldsymbol{x}_i)\boldsymbol{\psi}(\boldsymbol{x}_i)^\top$. Then, matrix inverse has to be computed only once for all dimensions:

$$(\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_d) = -(\boldsymbol{G} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d).$$

This significantly speeds up the computation particularly when the dimensionality d is high.

4.2 Multi-Task Learning

The above common-basis setup allows us to employ the *regularized multi-task* method [10], by regarding the estimation problem of $g_j^*(\boldsymbol{x})$ as the j-th task. The basic idea of regularized multi-task learning is that, if $g_j^*(\boldsymbol{x})$ and $g_{j'}^*(\boldsymbol{x})$ are similar to each other, the corresponding parameters $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_{j'}$ are imposed to be close to each other. This idea can be implemented in the regularization framework as

$$\min_{\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_d} \left[\sum_{j=1}^{d} \left(\boldsymbol{\theta}_j^\top \boldsymbol{G}^{(j)} \boldsymbol{\theta}_j + 2\boldsymbol{\theta}_j^\top \boldsymbol{h}_j + \lambda_j \boldsymbol{\theta}_j^\top \boldsymbol{\theta}_j \right) + \gamma \sum_{j,j'=1}^{d} \gamma_{j,j'} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}\|^2 \right],$$

where $\lambda_j > 0$ is the ordinary regularization parameter for the j-th task, $0 \leq \gamma_{j,j'} \leq 1$ is the similarity between the j-th task and the j'-th task, and $\gamma > 0$ controls the strength of this multi-task regularizer. A notable advantage of this regularization approach is that the solution can be obtained analytically. When the task similarity $\gamma_{j,j'}$ is unknown, task similarity and solutions may be iteratively learned. More specifically, starting from $\gamma_{j,j'} = 1$ for all $j, j' = 1, \dots, d$, the solutions $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d$ are computed. Then, task similarity is updated, e.g., by $\gamma_{j,j'} = \exp(-\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}\|^2)$ for $j, j' = 1, \dots, d$, and the solutions $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d$ are computed again.

4.3 Sparse Estimation

Instead of the ℓ_2-regularizer $\lambda\|\boldsymbol{\theta}_j\|^2$, the ℓ_1-regularizer $\lambda\|\boldsymbol{\theta}_j\|_1$ may be used to obtain a sparse solution [25]. The entire regularization path (i.e., the solutions for all $\lambda \geq 0$) can also be computed efficiently, based on the piece-wise linearity of the solution path with respect to λ [12].

4.4 Bregman Loss

The squared loss can be generalized to the *Bregman loss* [3]. More specifically, for f being a differentiable and strictly convex function and $C_j^{(f)} = \int f(g_j^*(\boldsymbol{x}))p^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$,

$$\begin{aligned} J_j^{(f)}(g_j) &= \int \left(f(g_j^*(\boldsymbol{x})) - f(g_j(\boldsymbol{x})) - f'(g_j(\boldsymbol{x}))(g_j^*(\boldsymbol{x}) - g_j(\boldsymbol{x})) \right) p^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - C_j^{(f)} \\ &= \int \left(-f(g_j(\boldsymbol{x})) + f'(g_j(\boldsymbol{x}))g_j(\boldsymbol{x}) \right) p^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int f'(g_j(\boldsymbol{x}))\partial_j p^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \int \left(-f(g_j(\boldsymbol{x})) + f'(g_j(\boldsymbol{x}))g_j(\boldsymbol{x}) + \partial_j f'(g_j(\boldsymbol{x})) \right) p^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \end{aligned}$$

where $f'(t)$ is the derivative of $f(t)$ with respect to t and the last equality follows again from integration by parts. The empirical approximation of $J_j^{(f)}$ is given as

$$\widehat{J}_j^{(f)}(g_j) = \frac{1}{n} \sum_{i=1}^{n} \left(-f(g_j(\boldsymbol{x}_i)) + f'(g_j(\boldsymbol{x}_i))g_j(\boldsymbol{x}_i) + \partial_j f'(g_j(\boldsymbol{x}_i)) \right).$$

When $f(t) = t^2$, the Bregman loss is reduced to the squared loss and we can recover the LSLDG criterion (1). On the other hand, $f(t) = -\log t$ gives the *Kullback-Leibler loss* [17], $f(t) = t\log t - (1+t)\log(1+t)$ gives the *logistic loss* [23], and $f(t) = (t^{1+\alpha} - t)/\alpha$ for $\alpha > 0$ gives the *power loss* [2]. Although each choice has its own specialty, e.g., the power loss possesses high robustness against outliers, the squared loss was shown to be endowed with the highest numerical stability in terms of the *condition number* [16].

4.5 Blurring Mean Shift

Fukunaga and Hostetler originally proposed a mean shift algorithm for updating not only the data points but also the density estimation at each iteration [11]. Later, this algorithm was named the *blurring mean shift* [4, 6]. Combined with the idea of the blurring mean shift, another possible algorithm for LSLDG clustering is to re-estimate the log-density gradient at each iteration for new data points. This algorithm hopefully works well as the blurring mean shift does [4].

5 Experiments

In this section, we demonstrate the usefulness of the proposed LSLDG method.

A MATLAB implementation of LSLDG and its clustering algorithm based on the fixed-point approach is available from

`http://sugiyama-www.cs.titech.ac.jp/~sugi/software/LSLDG/index.html`

5.1 Illustration of Log-Density Gradient Estimation

We first illustrate how LSLDG estimates log-density gradients using $n = 1,000$ samples drawn from $p(\boldsymbol{x})$, where either

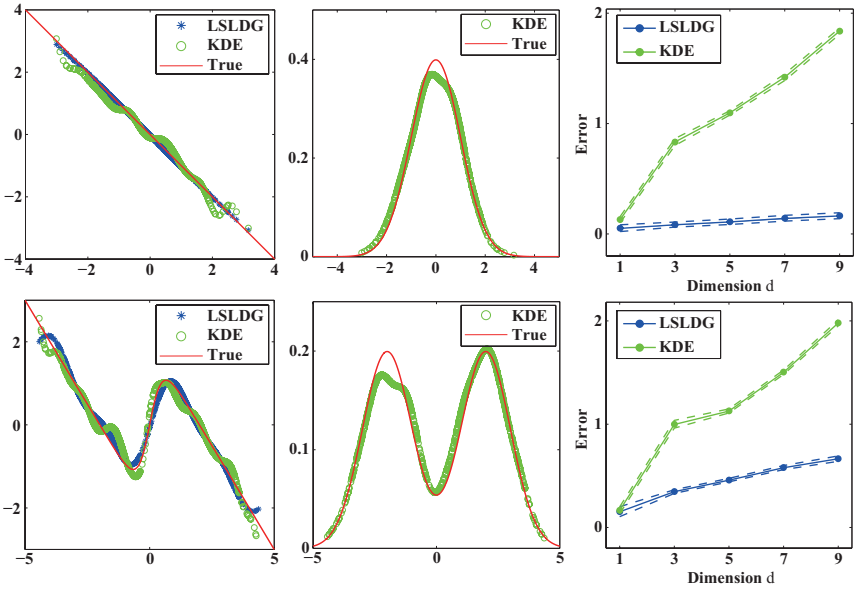
- $p(\boldsymbol{x})$ is the standard normal density, or
- $p(\boldsymbol{x})$ is a mixture of two Gaussians with means 2 and -2, variances 1 and 1, and mixing coefficients 0.5 and 0.5.

As described in Section 2.3, the Gaussian width σ and the regularization parameter λ are chosen by 5-fold cross-validation from the following candidate set:

$$\{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}, 10^{0.5}, 10^{1}\}. \tag{6}$$

We compare the performance of the proposed method with Gaussian KDE, where the Gaussian width is chosen by likelihood cross-validation from the same candidate set in (6).

The results for the Gaussian data are presented in the upper row of Figure 1. Figure 1(a) shows that LSLDG gives a nice smooth estimate, while the estimate by KDE is rather oscillating. Note that KDE still works well as a density estimator as illustrated in Figure 1(b). This clearly illustrates that a good density



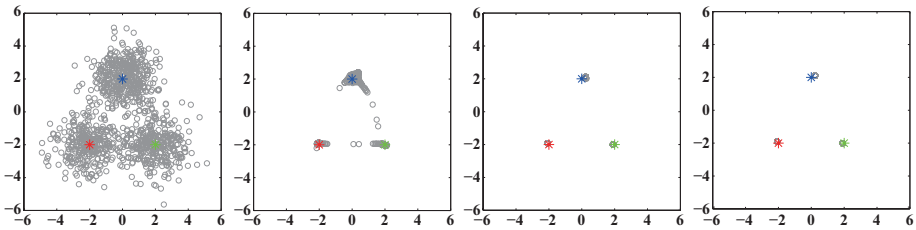
(a) Gradient estimation (b) Density estimation (c) Mean ℓ_2-norm error

Fig. 1. LSLDG vs. KDE for (upper row) Gaussian data and (lower row) data sampled from a mixture of two Gaussians. (a) Profiles of the true log-density gradient and its estimates obtained by LSLDG and KDE. (b) True and estimated densities by KDE. (c) Averages and standard deviations of mean ℓ_2-norm errors to the true log-density gradient as functions of input dimensionality over 100 runs.

estimate (obtained by KDE) does not necessarily yield a good estimate of the log-density gradient. We repeated this experiment 100 times and the mean ℓ_2-norm error to the true log-density gradient, $\frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{g}(\boldsymbol{x}_i) - \boldsymbol{g}^*(\boldsymbol{x}_i)\|$, is plotted in Figure 1(c) as a function of the input dimensionality. This shows that while the error of KDE increases sharply as a function of dimensionality, that of LSLDG increases only mildly. This implies that the advantage of directly estimating the log-density gradient is more prominent in high-dimensional cases. Similar tendencies can be observed also for the Gaussian mixture data in the lower row of Figure 1, where the added dimensions in the lower plot of Figure 1(c) simply follow the standard normal distribution.

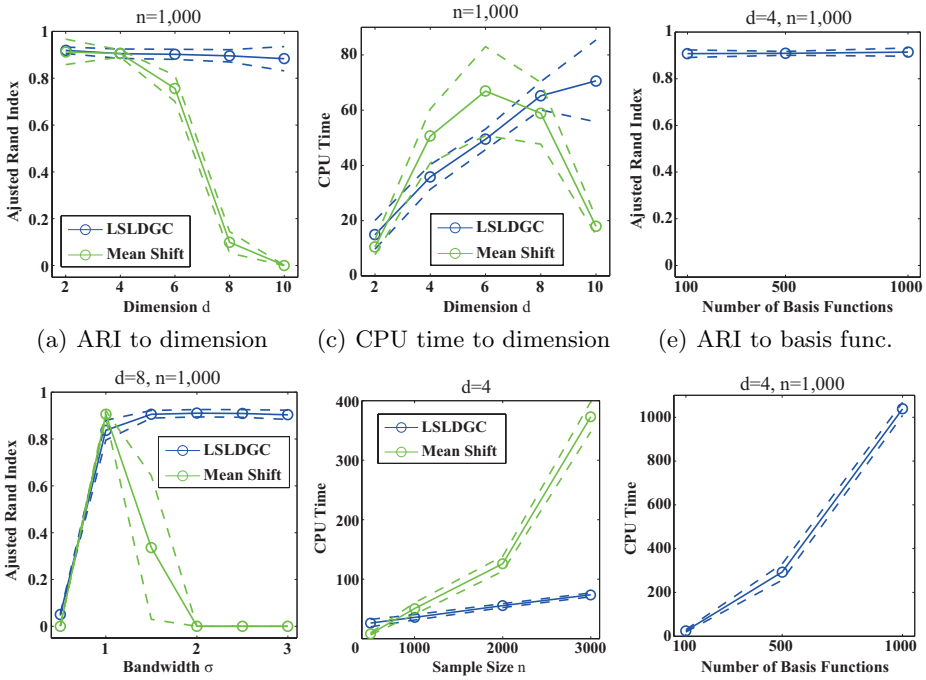
5.2 Illustration of Clustering

Next, we illustrate the behavior of LSLDG clustering on $1,000$ samples gathered from the mixture of three Gaussians whose means are $(0, 2)$, $(-2, -2)$, and $(2, -2)$, and covariance matrices are the identity matrix. The mixing coefficients are 0.4, 0.3, and 0.3. Figure 2 illustrates the transition of data samples over update iterations, showing that all points converge to the nearest modes within 47 iterations.



(a) Iteration = 0 (b) Iteration = 5 (c) Iteration = 15 (d) Iteration = 47

Fig. 2. Transition of data points toward the modes. The blue, red, and green symbols represent the three centers of the Gaussian mixture model.



(a) ARI to dimension (c) CPU time to dimension (e) ARI to basis func.

(b) ARI to bandwidth (d) CPU time to sample (f) CPU time to basis func.

Fig. 3. Means and standard deviations of clustering performance over 100 runs measured by ARI as functions of (a) dimensionality of data and (b) the Gaussian width (when dimensionality is 8). CPU time is also compared with respect to (c) dimensionality and (d) sample size. (e) ARI and (f) CPU time for LSLDG clustering are plotted as functions of the number of basis functions.

We compare the performance of the proposed method with the Gaussian mean shift [5, 6]. To investigate the effect of high dimensionality, further dimensions following the standard normal distribution are added to data points. We measure the clustering performance by the *adjusted Rand index* (ARI) [13], which takes the maximum value 1 when clustering is perfect.

ARI values are plotted as a function of input dimensionality in Figure 3(a) averaged over 100 runs. When the dimensionality of data is in the range of 2–4, both methods work very well. However, when the dimensionality is beyond 4, the performance of the Gaussian mean shift drops sharply. In contrast, for the proposed method, reasonably high ARI values are still attained even when the dimensionality is increased.

Figure 3(b) plots the ARI values for $d = 8$ when the Gaussian widths are changed. This shows that the proposed LSLDG clustering performs well for a wide range of Gaussian widths, while the ARI plot for the Gaussian mean shift is peaky. This implies that selection of Gaussian widths is much harder for the Gaussian mean shift than LSLDG clustering.

LSLDG clustering is also advantageous in terms of the computational costs. Figure 3(c) shows that CPU time of LSLDG clustering is almost the same as or shorter than that of the mean shift, when the ARI values for both methods are high enough. The shorter CPU time of the mean shift when the dimensionality is more than 8 comes from the fact that a smaller bandwidth is chosen; then the number of clusters is close to the number of kernels and thus the mean shift converges very quickly, although this choice is poor as a clustering method. With the same sample size, LSLDG clustering is much faster than the mean shift, as plotted in Figure 3(d). The speedup was brought by reducing the kernel centers, which was shown to significantly improve the computational costs without worsening the clustering performance, as depicted in Figures 3(e) and (f).

5.3 Image Discontinuity Preserving Smoothing and Image Segmentation

The mean shift has been successively applied to image discontinuity preserving smoothing and segmentation tasks [8, 24, 26]. Here, we investigate the performance of LSLDG clustering in those tasks.

As image data, we use the *Berkeley segmentation dataset* (BSD500) [1].[1] From one image, the information of color (three dimensions) and spatial positions (two dimensions) are extracted per pixel. Thus, the dimensionality of data is five, and the total number of samples is the same as the total number of pixels. As often assumed in the mean shift [8], for image data, we use the following mother basis function:

$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}^c - \boldsymbol{c}_i^c\|^2}{2\sigma_c^2}\right) \exp\left(-\frac{\|\boldsymbol{x}^s - \boldsymbol{c}_i^s\|^2}{2\sigma_s^2}\right), \tag{7}$$

where $\boldsymbol{x}^c$ and $\boldsymbol{x}^s$ denote the elements for colors and spatial positions in a data vector $\boldsymbol{x}$, respectively. $\boldsymbol{c}_i^c$ and $\boldsymbol{c}_i^s$ are the Gaussian centers. For the two Gaussian widths σ_c and σ_s, cross-validation is performed as in Section 2.3. In this experiment, we use a reduced image (11 by 16 or 16 by 11 pixels) as the Gaussian

[1] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`

Fig. 4. Examples of images after LSLDG clustering. The left-hand figure in each pair is the input image, and the right-hand one is the image after LSLDG clustering.

Table 1. Mean ARI values for 200 images. The numbers in the parentheses are standard deviations. The difference between the methods is statistically significant at level 1% by the t-test.

Mean Shift	LSLDGC
0.10(0.06)	**0.13(0.06)**

centers in (7). For the Gaussian mean shift, (7) is employed as a Gaussian kernel, and the two Gaussian widths are cross-validated based on the likelihood.

Six examples of color images after LSLDG clustering are shown in Figure 4. In the results, some of the segments, such as grass, are cleanly smoothed out, while the edges outlining the objects are preserved. These properties are similar to the results for the mean shift [8].

Next, to clarify the difference from the mean shift, we compare the performance measured by ARI. In this experiment, the input images are reduced to 81 by 121 (or 121 by 81) pixels. Since this benchmark dataset contains several ground truths per image, we simply computed the mean ARI value to all the ground truths.

The ARI values are summarized in Table 1, showing that LSLDG clustering outperforms the original mean shift on image segmentation.

5.4 Performance Comparison to Existing Clustering Methods

Finally, we compare LSLDG clustering to existing clustering methods using accelerometric sensor and speech data.

Table 2. Mean ARI for various methods over 100 runs. The standard deviations are indicated in the parentheses. The best method in terms of the average ARI and methods judged to be comparable to the best one by the t-test at the significance level 1% are described in boldface.

Accelerometry ($d = 5,\ n = 300,$ and $c = 3$)

KM	SC	Mean Shift	LSLDGC
0.50(0.03)	0.20(0.26)	0.51(0.05)	**0.61(0.13)**

Speech ($d = 50,\ n = 400,$ and $c = 2$)

KM	SC	Mean Shift	LSLDGC
0.00(0.00)	0.00(0.00)	0.00(0.00)	**0.13(0.02)**

For comparison, we employ K-means (KM) [18], spectral clustering (SC) [20, 19] with the Gaussian similarity, and Gaussian mean shift. Since the user has to set the number of clusters in advance for KM and SC, we set it at the true number of clusters in each dataset. For the Gaussian mean shift, the Gaussian width is chosen by likelihood cross-validation. For LSLDG, in this experiment, we modify the linear-in-parameter model as

$$g_j(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \psi_{i,j}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{\psi}_j(\boldsymbol{x}).$$

The main difference from the model introduced in Section 2.2 is that the coefficients θ_i do not depend on j, namely, the dimensionality of data. This modification considerably decreases the computational costs to higher dimensional data.

In this experiment, we used the following two datasets, where d denotes the dimensionality of data, n denotes the number of samples, and c denotes the number of true clusters:

1. *Accelerometry* ($d = 5, n = 300,$ and $c = 3$). The *ALKAN* dataset[2], which contains 3-axis (i.e., x-, y-, and z-axes) accelerometric data.
2. *Speech* ($d = 50, n = 400,$ and $c = 2$). An in-house speech dataset, which contains short utterance samples recorded from 2 male subjects speaking in French with sampling rate 44.1kHz.

The details of the two datasets can be seen in [22]. For each dataset, as preprocessing, the variance was normalized after centering in the element-wise manner.

The experimental results are described in Table 2. For the accelerometry dataset, LSLDG clustering shows the best performance among all the methods in the table. In addition to the superior performance, another advantage is that LSLDG clustering does not include any parameters which have to be manually tuned. On the other hand, KM and SC require the users to fix the number of

[2] `http://alkan.mns.kyutech.ac.jp/web/data.html`

clusters beforehand, which largely influences the clustering performance. Thus, LSLDG clustering would be easier to use in practice. For the speech dataset, LSLDG outperforms the existing clustering methods again (Table 2). Since the dimensionality of the dataset, $d = 50$, is much higher than the accelerometry dataset ($d = 5$), LSLDG seems to perform well on high-dimensional data, while the mean shift does not work well on high-dimensional data, as already indicated in Section 5.2.

6 Conclusions

In this paper, we developed a method to directly estimate the log-density gradient, and constructed a clustering algorithm on it. The proposed log-density gradient estimator can be regarded as a non-parametric extension of score matching [14, 21], and the proposed clustering algorithm can be regarded as an extension of the mean shift algorithm [6, 8, 11]. The key advantage compared to the original mean shift is that the proposed clustering method works well on high-dimensional data for which the mean shift works poorly. Furthermore, we showed experimentally that the proposed method outperforms existing clustering methods.

Acknowledgments H.S. was supported by KAKENHI 23120004. M.S. was supported by KAKENHI 25700022 and AOARD. A.H. was supported by the Academy of Finland CoE program, and the Japanese MIC project "Novel and innovative R&D making use of brain structures".

References

1. P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.
2. A. Basu, I. R. Harris, N. L. Hjort, and M. C. Jones. Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559, 1998.
3. L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.
4. M.Á. Carreira-Perpiñán. Fast nonparametric clustering with gaussian blurring mean-shift. In *ICML 2006*, pages 153–160. ACM, 2006.
5. M.Á. Carreira-Perpiñán. Gaussian mean-shift is an EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):767–776, 2007.
6. Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.
7. R. T. Collins. Mean-shift blob tracking through scale space. In *CVPR 2003*, volume 2, pages 234–240. IEEE, 2003.
8. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
9. D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR 2000*, volume 2, pages 142–149. IEEE, 2000.

10. T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2004)*, pages 109–117. ACM, 2004.
11. K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.
12. T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.
13. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
14. A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.
15. A. Hyvärinen. Some extensions of score matching. *Computational Statistics & Data Analysis*, 51(5):2499–2512, 2007.
16. T. Kanamori, T. Suzuki, and M. Sugiyama. Computational complexity of kernel-based density-ratio estimation: A condition number analysis. *Machine Learning*, 90(3):431–460, 2013.
17. S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.
18. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, CA, USA, 1967. University of California Press.
19. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*, pages 849–856, Cambridge, MA, USA, 2002. MIT Press.
20. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
21. B. Sriperumbudur, K. Fukumizu, A. Gretton, and A. Hyvärinen. Density estimation in infinite dimensional exponential families. *arXiv preprint arXiv:1312.3516*, 2013.
22. M. Sugiyama, G. Niu, M. Yamada, M. Kimura, and H. Hachiya. Information-maximization clustering based on squared-loss mutual information. *Neural Computation*, 26(1):84–131, 2014.
23. M. Sugiyama, T. Suzuki, and T. Kanamori. Density ratio matching under the Bregman divergence: A unified framework of density ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.
24. 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*, 37(5):1382–1389, 2007.
25. R. Tibshirani. Regression shrinkage and subset selection with the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.
26. J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *Computer Vision-ECCV 2004*, pages 238–249. Springer, 2004.