Regularized Multi-Task Learning for Multi-Dimensional Log-Density Gradient Estimation

Ikko Yamane
University of Tokyo, Japan
`yamane@ms.k.u-tokyo.ac.jp`

Hiroaki Sasaki
Nara Institute of Science and Technology, Japan
`hsasaki@is.naist.jp`

Masashi Sugiyama
University of Tokyo, Japan
`sugi@k.u-tokyo.ac.jp`

Abstract

Log-density gradient estimation is a fundamental statistical problem and possesses various practical applications such as clustering and measuring non-Gaussianity. A naive two-step approach of first estimating the density and then taking its log-gradient is unreliable because an accurate density estimate does not necessarily lead to an accurate log-density gradient estimate. To cope with this problem, a method to *directly* estimate the log-density gradient without density estimation has been explored, and demonstrated to work much better than the two-step method. The objective of this paper is to further improve the performance of this direct method in multi-dimensional cases. Our idea is to regard the problem of log-density gradient estimation in each dimension as a task, and apply *regularized multi-task learning* to the direct log-density gradient estimator. We experimentally demonstrate the usefulness of the proposed multi-task method in log-density gradient estimation and mode-seeking clustering.

Keywords

Multi-task learning, log-density gradient, mode-seeking

1 Introduction

Multi-task learning is a paradigm of machine learning for solving multiple related learning tasks simultaneously with the expectation that information brought by other related tasks can be mutually exploited to improve the accuracy (Caruana, 1997). Multi-task learning is particularly useful when one has many related learning tasks to solve but only few training samples are available for each task, which is often the case in many real-world problems such as therapy screening (Bickel et al., 2008) and face verification (Wang et al., 2009).

Multi-task learning has been gathering a great deal of attention, and extensive studies have been conducted both theoretically and experimentally (Thrun, 1996; Evgeniou and Pontil, 2004; Ando and Zhang, 2005; Zhang, 2013; Baxter, 2000). Thrun (1996) proposed the *lifelong learning framework*, which transfers the knowledge obtained from the tasks experienced in the past to a newly given task, and it was demonstrated to improve the performance of image recognition. Baxter (2000) defined a multi-task learning framework called *inductive bias learning*, and derived a generalization error bound. The semi-supervised multi-task learning method proposed by Ando and Zhang (2005) generates many auxiliary learning tasks from unlabeled data and seeks a good feature mapping for the target learning task. Among various methods of multi-task learning, one of the simplest and most practical approaches would be *regularized multi-task learning* (Evgeniou and Pontil, 2004; Evgeniou et al., 2005), which uses a regularizer that imposes the solutions of related tasks to be close to each other. Thanks to its generic and simple formulation, regularized multi-task learning has been applied to various types of learning problems such as regression and classification (Evgeniou and Pontil, 2004; Evgeniou et al., 2005). In this paper, we explore a novel application of regularized multi-task learning to the problem of *log-density gradient estimation* (Beran, 1976; Cox, 1985; Sasaki et al., 2014).

The goal of log-density gradient estimation is to estimate the gradient of the logarithm of an unknown probability density function using samples following it. Log-density gradient estimation has various applications such as clustering (Fukunaga and Hostetler, 1975; Cheng, 1995; Comaniciu and Meer, 2002; Sasaki et al., 2014), measuring non-Gaussianity (Huber, 1985) and other fundamental statistical topics (Singh, 1977).

Beran (1976) proposed a method for *directly* estimating gradients without going through density estimation, to which we refer as *least-squares log-density gradients* (LSLDG). This direct method was experimentally shown to outperform the naive one consisting of density estimation followed by log-gradient computation, and was demonstrated to be useful in clustering (Sasaki et al., 2014).

The objective of this paper is to estimate log-density gradients further accurately in multi-dimensional cases, which is still a challenging topic even using LSLDG. It is important to note that since the output dimensionality of the log-density gradient $\nabla \log p(\boldsymbol{x})$ is the same as its input dimensionality d, multi-dimensional log-density gradient estimation can be regarded as having multiple learning tasks if we regard estimation of each output dimension as a task. Based on this view, in this paper, we propose to apply regularized multi-task learning to LSLDG. We also provide a practically useful design of parametric models for successfully applying regularized multi-task learning to log-density gradient estimation. We experimentally demonstrate that the accuracy of LSLDG can be significantly improved by the proposed multi-task method in multi-dimensional log-density estimation problems and that a mode-seeking clustering method based on the proposed method outperforms other methods.

The organization of this paper is as follows: In Section 2, we formulate the problem of log-density gradient estimation and review LSLDG. Section 3 reviews the core idea of regularized multi-task learning. Section 4 presents our proposed log-density gradient estimator and algorithms for computing the solution. In Section 5, we experimentally demonstrate that the proposed method performs well on both artificial and benchmark data. Application to mode-seeking clustering is given in Section 6. Section 7 concludes this paper with potential extensions of this work.

2 Log-density gradient estimation

In this section, we formulate the problem of *log-density gradient estimation*, and then review LSLDG.

2.1 Problem formulation and a naive method

Suppose that we are given a set of samples, $\{\boldsymbol{x}_i\}_{i=1}^n$, which are independent and identically distributed from a probability distribution with unknown density $p(\boldsymbol{x})$ on $\mathbb{R}^d$. The problem is to estimate the gradient of the logarithm of the density $p(\boldsymbol{x})$ from $\{\boldsymbol{x}_i\}_{i=1}^n$:

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

where ∂_j denotes the partial derivative operator $\partial/\partial x^{(j)}$ for $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$.

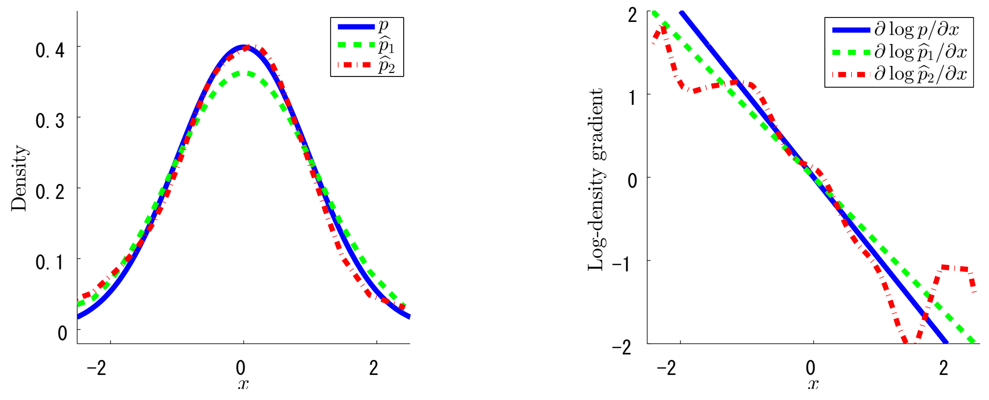
A naive method for estimating the log-density gradient is to first estimate the probability density, which is performed by, e.g., kernel density estimation (KDE) as

$$\widehat{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2\sigma^2}\right),$$

where $\sigma > 0$ denotes the Gaussian bandwidth, then to take the gradient of the logarithm of $\widehat{p}(\boldsymbol{x})$ as

$$\partial_j \log \widehat{p}(\boldsymbol{x}) = \frac{\partial_j \widehat{p}(\boldsymbol{x})}{\widehat{p}(\boldsymbol{x})}.$$

However, this two-step method does not work well because an accurate density estimate does not necessarily provide an accurate log-density gradient estimate. For example,



(a) Density estimation (b) Log-density gradient estimation

Figure 1: A comparison of two log-density gradient estimates based on density estimation. In (a), $\widehat{p}_2$ is a better estimate to the true density p than $\widehat{p}_1$, while in (b), $\nabla \log \widehat{p}_1$ is a better estimate to the true log-density gradient $\nabla \log p$ than $\nabla \log \widehat{p}_2$.

Figure 1 illustrates that a worse (or better) density estimate can produce a better (or worse) gradient estimate.

To overcome this problem, LSLDG, a single-step method which directly estimates the gradient without going through density estimation, was proposed (Beran, 1976; Cox, 1985; Sasaki et al., 2014), and has been demonstrated to experimentally work well. Next, we review LSLDG.

2.2 Direct estimation of log-density gradients

The basic idea of LSLDG is to directly fit a model $g_j(\boldsymbol{x})$ to the true log-density gradient $\partial_j \log p(\boldsymbol{x})$ under the squared loss:

$$
\begin{aligned}
R_j(g_j) &:= \int \left(g_j(\boldsymbol{x}) - \partial_j \log p(\boldsymbol{x})\right)^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \int \left(g_j(\boldsymbol{x}) - \frac{\partial_j p(\boldsymbol{x})}{p(\boldsymbol{x})}\right)^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - 2\int g_j(\boldsymbol{x})\partial_j p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + C_j \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - 2\int \left[g_j(\boldsymbol{x})p(\boldsymbol{x})\right]_{x^{(j)}=-\infty}^{x^{(j)}=\infty}\mathrm{d}\boldsymbol{x}^{(\backslash j)} + 2\int \partial_j g_j(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + C_j \\
&= \int g_j(\boldsymbol{x})^2 p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + 2\int \partial_j g_j(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + C_j,
\end{aligned}
$$

where $C_j := \int \frac{(\partial_j p(\boldsymbol{x}))^2}{p(\boldsymbol{x})}\mathrm{d}\boldsymbol{x}$ is a constant that does not depend on g_j, $\int(\cdot)\mathrm{d}\boldsymbol{x}^{(\backslash j)}$ denotes integration except for $x^{(j)}$, and the last deformation comes from *integration by parts* under the mild condition that $g_j(\boldsymbol{x})p(\boldsymbol{x}) \to 0$ as $|x^{(j)}| \to \infty$.

Then, the *LSLDG score* $J_j(g_j)$ is given as an empirical approximation to the risk $R_j(g_j)$ subtracted by C_j:

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

As $g_j(\boldsymbol{x})$, a linear-in-parameter model is used:

$$g_j(\boldsymbol{x}) = \boldsymbol{\theta}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}) = \sum_{k=1}^{b}\theta_j^{(k)}\psi_j^{(k)}(\boldsymbol{x}), \tag{2}$$

where $\theta_j^{(k)}$ is a parameter, $\psi_j^{(k)}(\boldsymbol{x})$ is a differentiable basis function, and b is the number of the basis functions. By substituting (2) into (1) and adding an ℓ_2-regularizer, we can analytically obtain the optimal solution $\widehat{\boldsymbol{\theta}}_j$ as

$$\begin{aligned}\widehat{\boldsymbol{\theta}}_j &= \mathop{\arg\min}_{\boldsymbol{\theta}_j}\left[\boldsymbol{\theta}_j^\top \boldsymbol{G}_j\boldsymbol{\theta}_j + 2\boldsymbol{h}_j^\top\boldsymbol{\theta}_j + \lambda_j\|\boldsymbol{\theta}_j\|^2\right]\\ &= -(\boldsymbol{G}_j + \lambda_j\boldsymbol{I}_b)^{-1}\boldsymbol{h}_j,\end{aligned}$$

where $\lambda_j \geq 0$ is the regularization parameter, $\boldsymbol{I}_b$ is the $b\times b$ identity matrix, and

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

Finally, an estimator of the log-density gradient is obtained by

$$\hat{g}_j(\boldsymbol{x}) := \hat{\boldsymbol{\theta}}_j^\top\boldsymbol{\psi}_j(\boldsymbol{x}).$$

It was experimentally shown that LSLDG produces much more accurate estimates of log-density gradients than the KDE-based gradient estimator and that the clustering method based on LSLDG performs well (Sasaki et al., 2014).

3 Regularized multi-task learning

In this section, we review a multi-task learning framework called *regularized multi-task learning* (Evgeniou and Pontil, 2004; Evgeniou et al., 2005), which is powerful and widely applicable to many machine learning methods.

Consider that we have T tasks of supervised learning as follows. The task t is to learn an unknown function $f_t^*(\boldsymbol{x})$ from samples of input-output pairs $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$, where $y_i^{(t)}$ is the output $f_t^*(\boldsymbol{x})$ with noise at the input $\boldsymbol{x} = \boldsymbol{x}_i^{(t)}$. When $f_t^*(\boldsymbol{x})$ is modeled by a parameterized function $f_t(\boldsymbol{x};\boldsymbol{\theta}_t)$, learning is performed by finding the parameter $\boldsymbol{\theta}_t$ which minimizes the empirical risk associated with some loss function $l(y, y')$:

$$\widehat{\boldsymbol{\theta}}_t = \mathop{\arg\min}_{\boldsymbol{\theta}_t}\frac{1}{n_t}\sum_{i=1}^{n_t} l(y_i^{(t)}, f_t(\boldsymbol{x}_t^{(t)};\boldsymbol{\theta}_t)) = \mathop{\arg\min}_{\boldsymbol{\theta}_t} J_t(\boldsymbol{\theta}_t),$$

where $J_t(\boldsymbol{\theta}_t) = \sum_{i=1}^{n_t} l(y_i^{(t)}, f_t(\boldsymbol{x}_t^{(t)}; \boldsymbol{\theta}_t))$.

In regularized multi-task learning, the objective function has regularization terms which impose every pair of parameters to be close to each other while $J_t(\boldsymbol{\theta}_t)$ are jointly minimized:

$$\sum_{t=1}^{T} J_t(\boldsymbol{\theta}_t) + \frac{1}{2}\gamma \sum_{t=1,t'=1}^{T} \gamma_{t,t'} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t'}\|^2,$$

where $\gamma \geq 0$ is the regularization parameter and $\gamma_{t,t'} \geq 0$ are the similarity parameters between the tasks t and t'.

It was experimentally demonstrated that the *multi-task support vector regression* (Evgeniou and Pontil, 2004; Evgeniou et al., 2005), performs better than the single-task counterpart (Vapnik et al., 1997) especially when the tasks are highly related each other.

4 Proposed method

In this section, we present our proposed method and algorithms.

4.1 Basic idea

Our goal in this paper is to improve the performance of LSLDG in *multi-dimensional* cases. For multi-dimensional input $\boldsymbol{x}$, the log-density gradient $\nabla \log p(\boldsymbol{x})$ has multiple output dimensions, meaning that its estimation actually consists of multiple learning tasks. Our basic idea is to apply regularized multi-task learning to solve these tasks simultaneously instead of learning them independently.

This idea is supported by the fact that the target functions of these tasks, $\partial_1 \log p(\boldsymbol{x}), \ldots, \partial_d \log p(\boldsymbol{x})$, are all derived from the same log-density $\log p(\boldsymbol{x})$, and thus they must be strongly related to each other. Under such strong relatedness, jointly learning them with sharing information with each other would improve estimation accuracy as has been observed in other existing multi-task learning work.

4.2 Regularized multi-task learning for least-squares log-density gradients (MT-LSLDG)

Here, we propose a method called *regularized multi-task learning for least-squares log-density gradients* (MT-LSLDG).

Our method MT-LSLDG is given by applying regularized multi-task learning to LSLDG. Specifically, we consider the problem of minimizing the following objective function:

$$\begin{aligned} J(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d) &= \sum_{j=1}^{d} J_j(g_j(\cdot; \boldsymbol{\theta}_j)) + \sum_{j=1}^{d} \lambda_j \|\boldsymbol{\theta}_j\|^2 + \frac{1}{2}\gamma \sum_{j,j'=1}^{d} \gamma_{j,j'} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}\|^2 \\ &= \sum_{j=1}^{d} \left(\boldsymbol{\theta}_j^\top \boldsymbol{G}_j \boldsymbol{\theta}_j + 2\boldsymbol{\theta}_j^\top \boldsymbol{h}_j + \lambda_j \|\boldsymbol{\theta}_j\|^2 \right) + \frac{1}{2}\gamma \sum_{j,j'=1}^{d} \gamma_{j,j'} \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}\|^2, \qquad (3) \end{aligned}$$

where the last term is the multi-task regularizer which imposes the parameters close to each other.

Denoting the minimizers of (3) by $\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_d$, the estimator $\widehat{\boldsymbol{g}}(\boldsymbol{x}) = (\widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_d(\boldsymbol{x}))^\top$ is given by, for $j = 1, \ldots, d$,

$$\widehat{g}_j(\boldsymbol{x}) = g_j(\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_j) = \widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}). \qquad (4)$$

We call this method *regularized multi-task learning for least-squares log-density gradients* (MT-LSLDG).

4.3 The design of the basis functions

The design of the basis functions $\boldsymbol{\psi}_j(\boldsymbol{x})$ in MT-LSLDG is crucial to enjoy the advantage of regularized multi-task learning. A simple design would be to use a common function $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}), \ldots, \phi^{(b)}(\boldsymbol{x}))$ for all $\boldsymbol{\psi}_j(\boldsymbol{x})$, that is, $\boldsymbol{\psi}_1(\boldsymbol{x}) = \cdots = \boldsymbol{\psi}_d(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})$. From (3) and (4), in this design, the multi-task regularizer promotes $g_j(\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_j)$ to be more close to each other so that

$$g_1(\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_1) \approx \cdots \approx g_d(\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_d).$$

However, it is inappropriate that all $g_j(\boldsymbol{x}; \widehat{\boldsymbol{\theta}}_j)$ are similar because the different true partial derivatives, say $\partial_j \log p(\boldsymbol{x})$ and $\partial_{j'} \log p(\boldsymbol{x})$ for $j \neq j'$, show different profiles in general.

To avoid this problem, we propose to use the partial derivatives of $\boldsymbol{\phi}(\boldsymbol{x})$ as basis functions:

$$\boldsymbol{\psi}_j(\boldsymbol{x}) = \partial_j \boldsymbol{\phi}(\boldsymbol{x}).$$

For this basis function design, it holds that

$$\widehat{\boldsymbol{g}}_j(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_j^\top \partial_j \boldsymbol{\phi}(\boldsymbol{x}) = \partial_j \widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\phi}(\boldsymbol{x}) \approx \partial_j \log p(\boldsymbol{x}).$$

Thus, $\widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\phi}(\boldsymbol{x})$ are approximations of the true log-density $\log p(\boldsymbol{x})$. Since the multi-task regularizer encourages the log-density estimates $\widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\phi}(\boldsymbol{x})$ to be more similar, this basis design would be reasonable.

As a specific choice of $\phi^{(k)}(\boldsymbol{x})$, we use a Gaussian kernel:

$$\begin{aligned}\psi_j^{(k)}(\boldsymbol{x}) &= \partial_j \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_k\|^2}{2\sigma^2}\right)\\ &= \frac{c_k^{(j)} - x^{(j)}}{\sigma^2}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_k\|^2}{2\sigma^2}\right),\end{aligned}$$

where $\boldsymbol{c}_k$ are the centers of the kernels, and $\sigma > 0$ is the Gaussian bandwidth.

4.4 Hyper-parameter tuning

As in LSLDG, the hyper-parameters, which are the ℓ_2-regularization parameters λ_j, the Gaussian bandwidth σ, and the multi-task parameters $\gamma, \gamma_{j,j'}$, can be cross-validated in MT-LSLDG. The procedure of the K-fold cross-validation is as follows: First, we randomly partition the set of training samples S_{tr} into K folds $F_1, \ldots, F_K$. Next, for each $k = 1, \ldots, K$, we estimate the log-density gradient using the samples in $S_{\mathrm{tr}} \setminus F_k$, which is denoted by $\hat{g}_j^{(k)}$, and then calculate the LSLDG scores for the samples in F_k as $J_{\mathrm{CV}}^{(k)}$:

$$J_{\mathrm{CV}}^{(k)} = \frac{1}{|F_k|}\sum_{\boldsymbol{x}\in F_k} \widehat{g}_j^{(k)}(\boldsymbol{x})^2 + \frac{2}{|F_k|}\sum_{\boldsymbol{x}\in F_k} \frac{\partial \widehat{g}_j^{(k)}(\boldsymbol{x})}{\partial x^{(j)}}.$$

We average these LSLDG scores to obtain the K-fold cross-validated LSLDG score:

$$J_{\mathrm{CV}} = \frac{1}{K}\sum_{k=1}^{K} J_{\mathrm{CV}}^{(k)}.$$

Finally, we choose the hyper-parameters that minimize J_{CV}. Throughout this paper, we set $K = 5$.

When the number of similarity parameters $\gamma_{j,j'}$ is large (i.e. the data dimensionality is high), cross-validation may be computationally inefficient. In this case, we may use the heuristic procedure which alternatingly updates the estimates $\widehat{\boldsymbol{\theta}}_j$ and the similarity parameters $\gamma_{j,j'}$ as described in Algorithm 1, where α is a hyper-parameter to be selected by cross-validation. In the update formula (5), the procedure determines the similarity parameter $\gamma_{j,j'}$ to be used in the next iteration depending on how close the estimated parameters $\widehat{\boldsymbol{\theta}}_j$ and $\widehat{\boldsymbol{\theta}}_{j'}$ are.

4.5 Optimization algorithms in MT-LSLDG

Here, we develop two algorithms for minimizing (3). One algorithm is to directly evaluate the analytic solution and the other is an iterative method based on block coordinate descent (Warga, 1963).

Algorithm 1 Similarity parameter tuning.

```
$\gamma_{j,j'} \leftarrow 1$ for every $j = 1, \dots, d; j' = 1, \dots, d$.
repeat
    With the current values of $\gamma_{j,j'}$, calculate the estimates as
        $(\widehat{\boldsymbol{\theta}}_1, \dots, \widehat{\boldsymbol{\theta}}_d) \leftarrow \arg\min_{(\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_d)} J(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d)$.
    for $j = 1, \dots, d; j' = 1, \dots, d$ do
        $\gamma_{j,j'} \leftarrow \exp(-\|\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}\|^2/\alpha^2)$.   (5)
    end for
until $\gamma_{j,j'}$ converges for every $j, = 1, \dots, d; j' = 1, \dots, d$.
```

4.5.1 Analytic solution

For simplicity, we assume the similarity parameters are symmetric: $\gamma_{j,j'} = \gamma_{j',j}$. Then, the objective function $J(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d)$ can be expressed as a quadratic function in terms of $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, \dots, \boldsymbol{\theta}_d^\top)^\top$ as

$$J(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top(\boldsymbol{G} + \boldsymbol{C} \otimes \boldsymbol{I}_b)\boldsymbol{\theta} + 2\boldsymbol{\theta}^\top \boldsymbol{h},$$

where

$$\boldsymbol{G} = \mathrm{diag}(\boldsymbol{G}_1, \dots, \boldsymbol{G}_d),\ \boldsymbol{h} = (\boldsymbol{h}_1^\top, \dots, \boldsymbol{h}_d^\top)^\top,$$
$$\boldsymbol{C} := \mathrm{diag}(\lambda_1, \dots, \lambda_d) + \gamma\, \mathrm{diag}\left(\sum_{j=1}^d \gamma_{1,j}, \dots, \sum_{j=1}^d \gamma_{d,j}\right) - \gamma\boldsymbol{\Gamma},$$

$[\boldsymbol{\Gamma}]_{j,j'} = \gamma_{j,j'}$, $\mathrm{diag}(\cdot, \dots, \cdot)$ is the block-diagonal matrix whose diagonal blocks are its arguments, and $\otimes$ denotes the Kronecker product. The minimizer $\widehat{\boldsymbol{\theta}}$ of $J(\boldsymbol{\theta})$ is analytically computed by

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -(\boldsymbol{G} + \boldsymbol{C} \otimes \boldsymbol{I}_b)^{-1}\boldsymbol{h}. \tag{6}$$

4.5.2 Block coordinate descent (BCD) method

Direct computation of the analytic solution (6) involves inversion of a $db \times db$ matrix. This may be not only expensive in terms of computation time but also infeasible in terms of memory space when the dimensionality d is very large.

Alternatively, we propose an algorithm based on block coordinate descent (BCD) (Warga, 1963). It is an iterative algorithm which only needs manipulation of a relatively small $b \times b$ matrix at each iteration. This alleviates the memory size requirement and hopefully reduces computation time if the number of iterations is not large.

Algorithm 2 Block coordinate descent (BCD) algorithm.

```
Initialize $\widetilde{\boldsymbol{\theta}}_1, \dots, \widetilde{\boldsymbol{\theta}}_d$.
repeat
    for $j = 1, \dots, d$ do
        $$\begin{aligned} \widetilde{\boldsymbol{\theta}}_j &\leftarrow \arg\min_{\boldsymbol{\theta}_j} J(\widetilde{\boldsymbol{\theta}}_1, \dots, \widetilde{\boldsymbol{\theta}}_{j-1}, \boldsymbol{\theta}_j, \widetilde{\boldsymbol{\theta}}_{j+1}, \dots, \widetilde{\boldsymbol{\theta}}_d) \\ &= \left(\boldsymbol{G}_j + \lambda_j \boldsymbol{I}_b + 2\gamma \sum_{j \neq j'} \gamma_{j,j'} \boldsymbol{I}_b\right)^{-1} \left(-\boldsymbol{h}_j + 2\gamma \sum_{j' \neq j} \gamma_{j,j'} \widetilde{\boldsymbol{\theta}}_{j'}\right). \end{aligned} \quad (7)$$
    end for
until $\widetilde{\boldsymbol{\theta}}_1, \dots, \widetilde{\boldsymbol{\theta}}_d$ converge.
```

A pseudo code of the algorithm is shown in Algorithm 2. At each update (7) in the algorithm, only one vector $\widetilde{\boldsymbol{\theta}}_j$ is optimized in a closed-form while fixing the other parameters $\widetilde{\boldsymbol{\theta}}_{j'}$ $(j' \neq j)$. The update (7) only requires computing the inverse of a $b \times b$ matrix, which seems to be computationally advantageous over evaluating the analytic solution in terms of the computation cost and memory size requirement.

Another important technique to reduce the overall computation time is to use *warm start* initialization: when the optimal value of γ is searched for by cross-validation, we may use the solutions $\widetilde{\boldsymbol{\theta}}_1, \dots, \widetilde{\boldsymbol{\theta}}_d$ obtained with γ as initial values for another γ.

5 Experiments on log-density gradient estimation

In this section, we illustrate the behavior of the proposed method and experimentally investigate its performance.

5.1 Experimental setting

In each experiment, training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ and test samples $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ are drawn independently from an unknown density $p(\boldsymbol{x})$. We estimate $\nabla \log p(\boldsymbol{x})$ from the training samples, and then evaluate the estimation performance by the *test score*

$$J_{\mathrm{te}}(\widehat{\boldsymbol{g}}) = \sum_{j=1}^{d} \left[\frac{1}{n'} \sum_{i'=1}^{n'} \widehat{g}_j(\boldsymbol{x}'_{i'})^2 + \frac{2}{n'} \sum_{i'=1}^{n'} \partial_j \widehat{g}_j(\boldsymbol{x}'_{i'}) \right],$$

where $\widehat{\boldsymbol{g}}(\boldsymbol{x}) = (\widehat{g}_1(\boldsymbol{x}), \dots, \widehat{g}_d(\boldsymbol{x}))^\top$ is an estimated log-density gradient. This score is an empirical approximation of the expected squared loss of $\widehat{\boldsymbol{g}}(\boldsymbol{x})$ over the test samples without the constant C_j (see Section 2.2), and a smaller score means a better estimate.

We compare the following three methods:

- The multi-task LSLDG (MT-LSLDG): our method proposed in Section 4.
- The single-task LSLDG (S-LSLDG): the existing method (Beran, 1976; Cox, 1985) reviewed in Section 2.2. This method agrees with MT-LSLDG at $\gamma = 0$.
- The common-parameter LSLDG (C-LSLDG): LSLDG with common parameters $\boldsymbol{\theta}' = \boldsymbol{\theta}_1 = \cdots = \boldsymbol{\theta}_d$ learned simultaneously. The solution is given as

$$\begin{aligned}\widehat{\boldsymbol{\theta}'} &= \arg\min_{\boldsymbol{\theta}'} \left[\boldsymbol{\theta}'^{\top} \sum_{j=1}^{d} \boldsymbol{G}_j \boldsymbol{\theta}' + 2 \sum_{j=1}^{d} \boldsymbol{h}_j^{\top} \boldsymbol{\theta}' + \lambda \|\boldsymbol{\theta}'\|^2 \right] \\ &= - \left(\sum_{j=1}^{d} \boldsymbol{G}_j + \lambda \boldsymbol{I}_b \right)^{-1} \sum_{j=1}^{d} \boldsymbol{h}_j,\end{aligned}$$

  where $\lambda \geq 0$ is the ℓ_2-regularization parameter. This method agrees with MT-LSLDG at the limit $\gamma \to \infty$.

In all the methods, we set the number of basis functions as $b = \min\{50, n\}$, and randomly choose the kernel centers $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_b$ uniformly from training samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$ without replacement. For hyper-parameters, we use the common ℓ_2-regularization parameter λ and bandwidth parameter σ among all the dimensions, $\lambda_1 = \cdots = \lambda_d = \lambda$ and $\sigma_1 = \cdots = \sigma_d = \sigma$. We also set all the similarity parameters as $\gamma_{j,j'} = 1$, which assumes that all dimensions are equally related to each other. In order to examine whether this assumption is reasonable, we experimentally compare MT-LSLDG with this assumption and that with the similarity parameter tuning (Algorithhm 1) in Section 5.2.

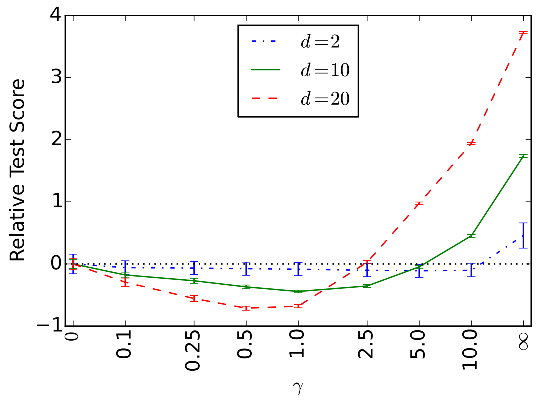
5.2 Artificial data

We conduct numerical experiments on artificial data to investigate the basic behavior of MT-LSLDG. As data density $p(\boldsymbol{x})$, we consider the following two cases:

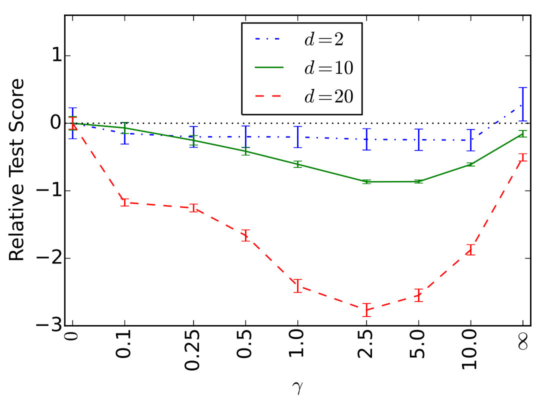
- Single Gaussian: The d-dimensional Gaussian density whose mean is $\mathbf{0}$ and whose covariance matrix is the diagonal matrix with the first half of the diagonal elements are 1 and the others are 5.
- Double Gaussian: A mixture of two d-dimensional Gaussian densities with mean zero and $(5, 0, \ldots, 0)^{\top}$ and identity covariance matrix. The mixing coefficients are 1/2.

The dimensionality d and sample size n are specified later.

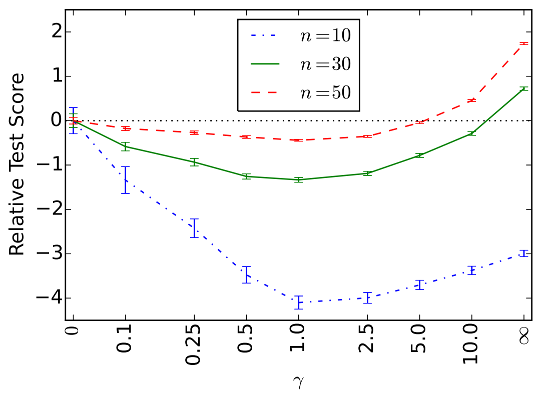
First, we investigate whether MT-LSLDG improves the estimation accuracy of LSLDG at appropriate γ. We prepare datasets with different dimensionalities $d = 2, 10, 20$ and sample sizes $n = 10, 30, 50$. MT-LSLDG is applied to the datasets at each $\gamma \in \{0, 0.1, 0.25, 0.5, 1, 2.5, 5, 10, \infty\}$. The Gaussian bandwidth σ and the ℓ_2-regularization parameter λ are chosen by 5-fold cross-validation as described in Section 4 from the



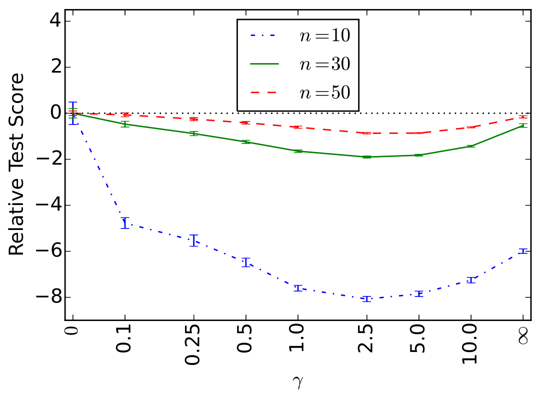
(a) Single Gaussian ($n = 30$)



(b) Double Gaussian ($n = 30$)



(c) Single Gaussian ($d = 10$)



(d) Double Gaussian ($d = 10$)

Figure 2: Average (and standard errors) of *relative test scores* over 100 runs. The relative test scores refer to test scores from which the test score of S-LSLDG is subtracted. The black dotted lines indicate the relative score zero.

candidate lists $\{10^{-1}, 10^{-0.25}, 10^{0.5}, 10^{1.25}, 10^{2}\}$ and $\{10^{-3}, 10^{-2}, 10^{-1}\}$, respectively. The solution of MT-LSLDG is computed analytically as in (6).

The results are plotted in Figure 2. In the figure, the relative test score is defined as the test score from which the test score of S-LSLDG is subtracted, and thus negative relative scores indicate that MT-LSLDG improved the performance of S-LSLDG. When $d = 2$, MT-LSLDG does not improve the performance for any γ values (Figure 2(a) and 2(b)). However, for higher-dimensional data, the performance is improved at appropriate γ values (e.g., $\gamma = 0.5$ for $d = 20$ in Figure 2(a) and $\gamma = 2.5$ for $d = 20$ in Figure 2(b)). Similar improvement is observed also for smaller sample size (e.g., $n = 10$ and $n = 30$) in Figure 2(c) and Figure 2(d).

These results confirm that MT-LSLDG improves the performance of S-LSLDG at an appropriate γ value when data is relatively high-dimensional and the sample size is small. Since such γ is usually unknown in advance, we need to find a reasonable value in practice.

Table 1: Averages (and standard errors) of test scores on the artificial data with cross-validation over 100 runs. MT-LSLDG-T in the table refers to MT-LSLDG with similarity parameter tuning by Algorithm 1. In each row, the best and comparable to the best scores in terms of paired t-test with significance level 5% are emphasized in bold face.

Density	n	MT-LSLDLG	MT-LSLDG-T	S-LSLDG	C-LSLDG
Single Gaussian	10	**−2.87** (0.22)	−2.33 (0.37)	0.37 (0.31)	**−2.58** (0.07)
	30	**−5.34** (0.04)	**−5.38** (0.02)	−4.97 (0.08)	−3.29 (0.03)
	50	**−5.63** (0.02)	**−5.64** (0.01)	−5.55 (0.02)	−4.13 (0.04)
Double Gaussian	10	**−6.83** (0.14)	**−6.80** (0.17)	1.01 (0.54)	−5.02 (0.12)
	30	**−8.45** (0.03)	**−8.45** (0.07)	−7.63 (0.10)	−7.84 (0.04)
	50	−8.67 (0.02)	**−8.71** (0.03)	−8.29 (0.10)	−8.48 (0.02)

Density	d	MT-LSLDG	MT-LSLDG-T	S-LSLDG	C-LSLDG
Single Gaussian	2	**0.20** (0.19)	0.21 (0.15)	**−0.11** (0.15)	**0.09** (0.15)
	10	**−5.34** (0.04)	**−5.38** (0.02)	−4.97 (0.08)	−3.29 (0.03)
	20	**−10.77** (0.03)	**−10.76** (0.04)	−9.98 (0.13)	−6.39 (0.01)
Double Gaussian	2	**0.54** (0.22)	**0.51** (0.25)	**0.50** (0.27)	**0.19** (0.22)
	10	**−8.45** (0.03)	**−8.45** (0.07)	−7.63 (0.10)	−7.84 (0.04)
	20	**−16.88** (0.14)	**−17.06** (0.125)	−14.90 (0.10)	−15.26 (0.06)

Next, we investigate whether an appropriate γ value can be chosen by cross-validation. In this experiment, the cross-validation method in Section 4.3 is performed to choose γ as well. The candidates of γ is $\{0, 0.1, 0.25, 0.5, 1, 2.5, 5, 10, \infty\}$. The other experimental settings such as the data generation and all the LSLDGs are the same as in the last experiment except that we also run MT-LSLDG with similarity parameter tuning by Algorithm 1 with cross-validated α. The candidate list for α is $\{10^{-1}, 10^{-1/4}, 10^{2/4}, 10^{5/4}, 10^{2}\}$.

Table 1 shows that MT-LSLDG improves the performance especially when the dimensionality of data is relatively high and the sample size is small. These results indicate that the proposed cross-validation method allows us to choose a reasonable γ value. In most cases of the experiments, MT-LSLDG with $\gamma_{j,j'} = 1$ gives estimation accuracy comparable to that with the similarity parameter tuning procedure. In the remaining experiments, we use cross-validated γ and the fixed $\gamma_{j,j'} = 1$.

5.3 Benchmark data

In this section, we demonstrate the usefulness of MT-LSLDG in gradient estimation on various benchmark datasets. This experiment uses some IDA benchmark datasets (Rätsch et al., 2001) and UCI benchmark datasets (Catlett, 1991; Kaya et al., 2012; Lucas et al., 2013; Tüfekci, 2014; Lichman, 2013). All the datasets are standardized in advance.

For MT-LSLDG, the hyper-parameters σ, λ and γ are chosen by cross-validation. The candidate lists are $\sigma \in \{0.1, 0.25, 0.5, 1, 2.5, 5, 10\}$, $\lambda \in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ and $\gamma \in$

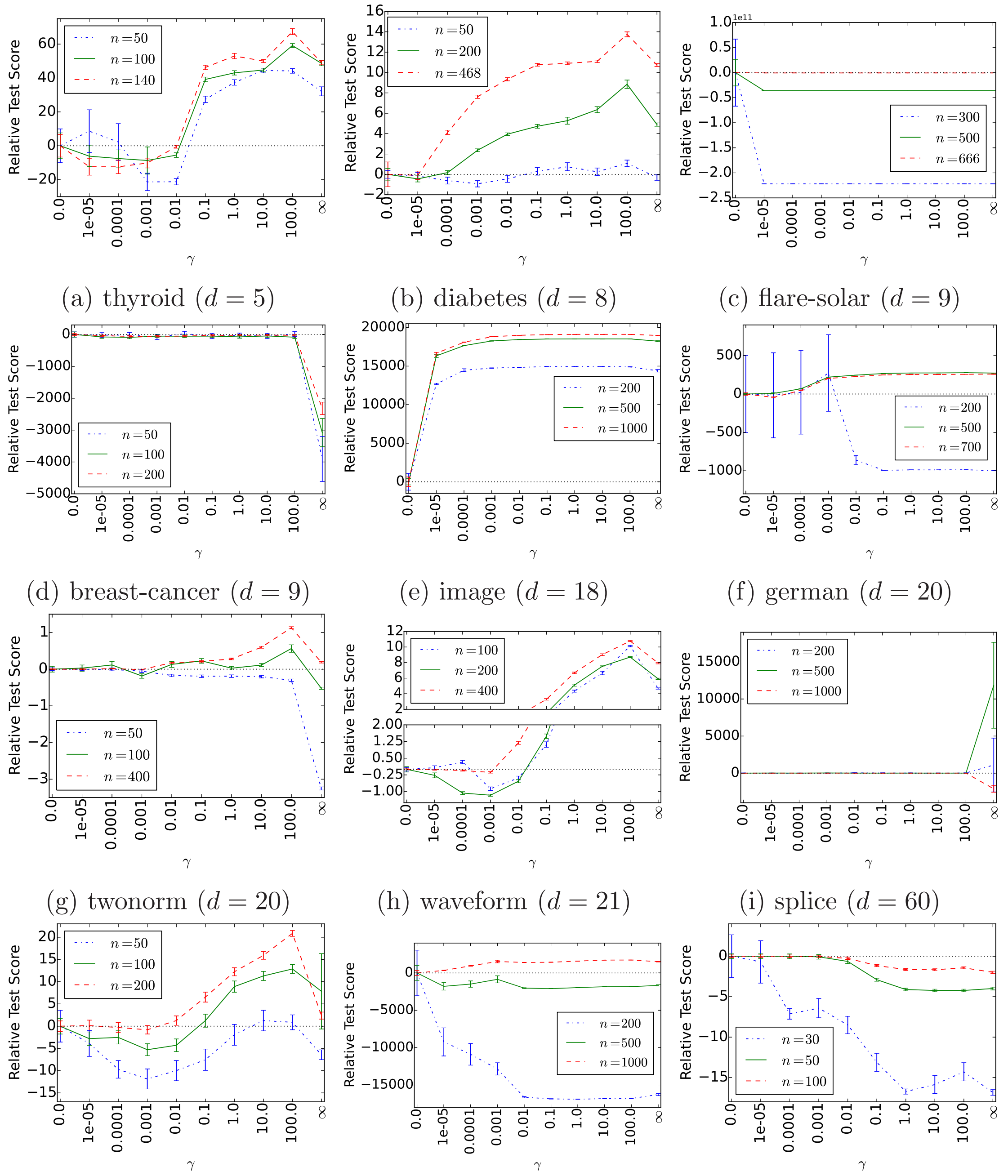
Table 2: Averages (and standard errors) of the test scores on the benchmark datasets. In each dataset, the best and comparable to the best scores in terms of paired t-test with significance level 5% are emphasized in bold face. The number of trials is 20 for the image and splice dataset, and is 100 for the other datasets.

Dataset (d, n)	MT-LSLDG	S-LSLDG	C-LSLDG
thyroid (5, 140)	$\mathbf{-1.076 \times 10^2}$	$\mathbf{-1.083 \times 10^2}$	-5.149×10
	(0.066×10^2)	(0.065×10^2)	(0.012×10)
CCPP (5, 200)	$\mathbf{-3.661 \times 10}$	$\mathbf{-3.585 \times 10}$	-3.232×10
	(0.120×10)	(0.125×10)	(0.187×10)
diabetes (8, 468)	$\mathbf{-2.240 \times 10}$	$\mathbf{-2.211 \times 10}$	-1.510×10
	(0.029×10)	(0.034×10)	(0.008×10)
flare-solar (9, 666)	$\mathbf{-1.341 \times 10^7}$	6.626×10^8	$\mathbf{-1.342 \times 10^7}$
	(0.021×10^7)	(3.251×10^8)	(0.021×10^7)
breast-cancer (9, 200)	$\mathbf{-2.535 \times 10^3}$	-2.169×10^2	$\mathbf{-2.535 \times 10^3}$
	(0.195×10^3)	(0.294×10^2)	(0.195×10^3)
shuttle (9, 1000)	$\mathbf{-2.664 \times 10^3}$	$\mathbf{-2.974 \times 10^3}$	-1.063×10^3
	(0.321×10^3)	(0.109×10^3)	(0.038×10^3)
image (18, 1300)	$\mathbf{-2.993 \times 10^3}$	$\mathbf{1.020 \times 10^4}$	-3.289×10^3
	(0.541×10^3)	(1.246×10^4)	(0.027×10^3)
popfailures (18, 50)	$\mathbf{-2.110 \times 10^2}$	-2.067×10^2	-2.108×10^2
	(0.002×10^2)	(0.003×10^2)	(0.002×10^2)
german	$\mathbf{-3.042 \times 10^2}$	$\mathbf{-2.960 \times 10^2}$	-1.999×10
	(0.140×10^2)	(0.092×10^2)	(0.373×10)
twonorm (20, 400)	$\mathbf{-2.233 \times 10}$	-2.232×10	-2.213×10
	(0.001×10)	(0.001×10)	(0.002×10)
waveform (21, 400)	$\mathbf{-4.332 \times 10}$	-4.321×10	-3.526×10
	(0.003×10)	(0.002×10)	(0.010×10)
splice (60, 1000)	$\mathbf{-2.484 \times 10^3}$	-6.027×10	$\mathbf{-2.382 \times 10^3}$
	(0.379×10^3)	(0.345×10)	(0.393×10^3)

$\{0, 10^{-5}, 10^{-4}, \ldots, 10^1, 10^2, \infty\}$, respectively. For S-LSLDG and C-LSLDG, the candidate lists of σ and λ are the same as MT-LSLDG. The solution of MT-LSLDG is computed by the BCD algorithm described in Section 4.5.2.

The results are presented in Table 2. MT-LSLDG significantly improves the performance of either S-LSLDG or C-LSLDG on most of the datasets.

We also run MT-LSLDG for three different samples sizes n for each of the datasets with changing γ from 0 to ∞. Figure 3 summarizes the results of the experiments. The plots show that performance highly depends on γ, and that the best γ differs from one dataset to another, which means that it is very important to select a good γ by cross-validation. Also, we can see that the blue plot, which is that for the smallest n, shows the largest improvement over S-LSLDG in most of the datasets. This implies that MT-LSLDG is



(a) thyroid ($d = 5$) (b) diabetes ($d = 8$) (c) flare-solar ($d = 9$)

(d) breast-cancer ($d = 9$) (e) image ($d = 18$) (f) german ($d = 20$)

(g) twonorm ($d = 20$) (h) waveform ($d = 21$) (i) splice ($d = 60$)

(j) CCPP ($d = 5$) (k) shuttle ($d = 9$) (l) popfailures ($d = 18$)

Figure 3: Average (and standard errors) of relative test scores on the IDA datasets and the other real datasets. The relative test scores refer to test scores from which the test score of S-LSLDG is subtracted. The black dotted lines indicate the relative score zero.

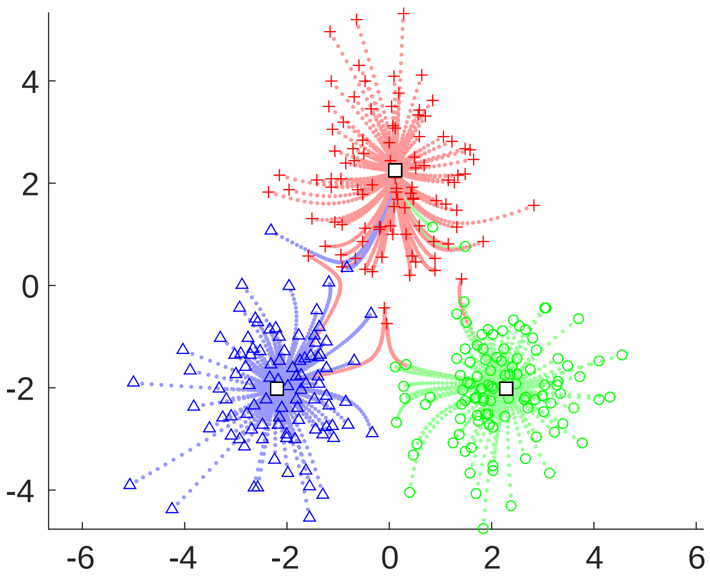


Figure 4: Transition of data points during a mode-seeking process. Data samples are drawn from a mixture of Gaussians, and the data points sampled from the same Gaussian component are specified by the same color (red, green, or blue) and marker (plus symbol, circle, or triangle). White squares indicate the points to which data points converged.

advantageous especially when the sample size is small.

6 Application to mode-seeking clustering

In this section, we apply MT-LSLDG to mode-seeking clustering and experimentally demonstrate its usefulness.

6.1 Mode-seeking clustering

A practical application of log-density gradient estimation is *mode-seeking clustering* (Fukunaga and Hostetler, 1975; Cheng, 1995; Comaniciu and Meer, 2002; Sasaki et al., 2014). Mode-seeking clustering methods update each data point toward a nearby mode by gradient ascent, and assign the same clustering label to the data points which converged to the same mode (Figure 4). Their notable advantage is that we need not specify the number of clusters in advance. Mode-seeking clustering has been successfully applied to a variety of real world problems such as object tracking (Comaniciu et al., 2000), image segmentation (Comaniciu and Meer, 2002; Sasaki et al., 2014), and line edge detection in images (Bandera et al., 2006).

In mode-seeking, the essential ingredient is the gradient of the data density. To estimate the gradients, *mean shift clustering* (Fukunaga and Hostetler, 1975; Cheng, 1995;

Comaniciu and Meer, 2002), which is one of the most popular mode-seeking clustering methods, employs the two-step method of first estimating the data density by kernel density estimation and then taking its gradient. However, as we mentioned earlier, this two-step method does not work well since accurately estimating the density does not necessarily lead to an accurate estimate of the gradient.

In order to overcome this problem, *LSLDG clustering* (Sasaki et al., 2014) adopted LSLDG instead of the two-step method. Sasaki et al. (2014) also provided a practically useful fixed-point algorithm for mode-seeking as in mean shift clustering (Cheng, 1995): When the partial derivative of a vector of Gaussian kernels $\boldsymbol{\psi}_j(\boldsymbol{x}) = \partial_j \boldsymbol{\phi}(\boldsymbol{x})$ is used as the vector of basis functions, the model $g_j(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}_j^\top \boldsymbol{\psi}_j(\boldsymbol{x})$ can be transformed as

$$\begin{aligned}
\widehat{g}_j(\boldsymbol{x}) &= \sum_{k=1}^{b} \widehat{\theta}_j^{(k)} \frac{c_k^{(j)} - x^{(j)}}{\sigma^2} \phi^{(k)}(\boldsymbol{x}) \\
&= \frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\theta}_j^{(k)} c_k^{(j)} \phi^{(k)}(\boldsymbol{x}) - \frac{x^{(j)}}{\sigma^2} \sum_{k=1}^{b} \widehat{\theta}_j^{(k)} \phi^{(k)}(\boldsymbol{x}) \\
&= \left[\frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\theta}_j^{(k)} \phi^{(k)}(\boldsymbol{x}) \right] \left[\frac{\sum_{k'=1}^{b} \widehat{\theta}_j^{(k')} c_{k'}^{(j)} \phi^{(k')}(\boldsymbol{x})}{\sum_{k'=1}^{b} \widehat{\theta}_j^{(k')} \phi^{(k')}(\boldsymbol{x})} - x^{(j)} \right],
\end{aligned}$$

where we assume that $\frac{1}{\sigma^2} \sum_{k=1}^{b} \widehat{\theta}_j^{(k)} \phi^{(k)}(\boldsymbol{x})$ is nonzero. Setting $\widehat{g}_j(\boldsymbol{x})$ to zero yields a fixed-point update formula as

$$x^{(j)} \leftarrow \frac{\sum_{k'=1}^{b} \widehat{\theta}_j^{(k')} c_{k'}^{(j)} \phi^{(k')}(\boldsymbol{x})}{\sum_{k'=1}^{b} \widehat{\theta}_j^{(k')} \phi^{(k')}(\boldsymbol{x})}.$$

It has been experimentally shown that LSLDG clustering performs significantly better than mean-shift clustering (Sasaki et al., 2014).

Here, we apply MT-LSLDG to LSLDG clustering and investigate if the performance is improved in mode-seeking clustering as well for relatively high-dimensional data.

6.2 Experiments

Next, we conduct numerical experiments for mode-seeking clustering.

6.2.1 Experimental setting

We apply the following four clustering methods to various datasets:

- MT-LSLDGC: LSLDG clustering with MT-LSLDG.
- S-LSLDGC: LSLDG clustering with S-LSLDG (Sasaki et al., 2014).
- C-LSLDGC: LSLDG clustering with C-LSLDG (Sasaki et al., 2014).

Table 3: Averages (and standard errors) of ARIs on artificial data. In each row, the best and comparable to the best ARI in terms of unpaired t-test with significance level 5% is emphasized in bold face. The number of trials is 100.

d	MT-LSLDGC	S-LSLDGC	C-LSLDGC	Mean-shift
2	**0.992** (0.035)	**0.973** (0.125)	**0.992** (0.036)	**0.984** (0.044)
10	**0.993** (0.004)	**0.994** (0.003)	**0.994** (0.004)	0.042 (0.022)
15	**0.983** (0.023)	**0.982** (0.054)	0.877 (0.217)	0.000 (0.000)
20	**0.827** (0.190)	0.586 (0.208)	0.716 (0.352)	0.036 (0.037)

- Mean-shift: mean shift clustering (Comaniciu and Meer, 2002).

For MTL-, S-, and C-LSLDG, all the hyper-parameters are cross-validated as described in Section 4.3, and for mean-shift, log-likelihood cross-validation is used.

We evaluate the clustering performance by the *adjusted Rand index* (ARI) (Hubert and Arabie, 1985). ARI gives one to the perfect clustering assignment and zero on average to a random clustering assignment. A larger ARI value means a better clustering result.

6.2.2 Artificial data

First, we conduct experiments on artificial data. The density of the artificial data is a mixture of three d-dimensional Gaussian densities with means $(0, 2, 0, \ldots, 0)$, $(-2, -2, 0, \ldots, 0)$, and $(2, -2, 0, \ldots, 0)$, covariance matrices $\frac{1}{\sqrt{2\pi}}\boldsymbol{I}_d$, and mixing coefficients 0.4,0.3,0.3. The candidate lists of the hyper-parameters are the following: $\sigma \in \{10^{-1}, 10^{-7/9}, 10^{-5/9}, \ldots, 10^{5/9}, 10^{7/9}, 10^{1}\}$, $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and, $\gamma \in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, \infty\}$.

The results are shown in Table 3. We can see that MT-LSLDGC performs well especially for the largest dimensionality $d = 20$.

6.2.3 Real data

Next, we perform clustering on real data. The following three datasets are used:

- Accelerometry data: 5-dimensional data used in (Hachiya et al., 2012) for human activity recognition extracted from mobile sensing data available from `http://alkan.mns.kyutech.ac.jp/web/data`. The number of classes is 3. In each run of experiment, we use randomly chosen 100 samples from each class. The total number of samples is 300.

- Vowel data: 10-dimensional data of recorded British English vowel sounds available from `https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data)`. The number of classes is 11 In each run of experiment, we use randomly chosen 500.

Table 4: Averages (and standard errors) of ARIs on real data. In each row, the best and comparable to the best ARI in terms of paired t-test with significance level 5% is emphasized in bold face. The number of trials is 100 for the accelerometry data and the sat-image data, and is 20 for the speech data.

dataset (d, n)	MT-LSLDGC	S-LSLDGC	C-LSLDGC	Mean-shift
accelerometry (5, 300)	0.40 (0.01)	**0.53** (0.02)	0.24 (0.01)	0.26 (0.04)
vowel (10, 500)	**0.15** (0.00)	**0.15** (0.00)	**0.15** (0.00)	0.04 (0.00)
sat-image (36, 2000)	**0.48** (0.00)	0.43 (0.01)	0.35 (0.00)	0.00 (0.00)
speech (50, 400)	**0.17** (0.02)	0.00 (0.00)	0.15 (0.01)	0.00 (0.00)

- Sat-image data: 36-dimensional multi-spectral satellite image available from `https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)`. The number of classes is 6. In each run of experiment, we use randomly chosen 2000 samples.

- Speech data: 50-dimensional voice data by two French speakers (Sugiyama et al., 2014). The number of classes is 2. In each run of experiment, we use randomly chosen 200 samples from each class. The total number of samples is 400.

For MT-LSLDG, the hyper-parameters are cross-validated using the candidates, $\sigma \in \{10^{-1}, 10^{-6/9}, 10^{-3/9}, \ldots, 10^{12/9}, 10^{15/9}, 10^{2}\}$, $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ and $\gamma \in \{10^{-5}, 10^{-4}, \ldots, 10^{1}, 10^{2}\}$, except that we use relatively small candidate lists $\sigma \in \{0.5, 1, 2.5, 5, 10\}$, $\lambda \in \{0.003, 0.01, 0.1, 1\}$ and $\gamma \in \{0.1, 1, 10\}$ for the speech data since it has large dimensionality and optimization is computationally expensive. For S- and C-LSLDG, we used the same candidates of MT-LSLDG for σ and λ. For mean shift clustering, the Gaussian kernel is employed in KDE, and the bandwidth parameter in the kernel is selected by 5-fold cross-validation with respect to the log-likelihood of the density estimate from the same candidates of MT-LSLDG for σ.

The results are shown in Table 4. For the accelerometry data whose dimensionality is only five, S-LSLDGC gives the best performance and MT-LSLDGC does not improve the performance, although MT-LSLDGC performs better than C-LSLDGC.

On the other hand, for the higher-dimensional dataset, the vowel data, the sat-image data, and the speech data, the performance of MT-LSLDGC is the best or comparable to the best. These results indicate that MT-LSLDG is a promising method in mode-seeking clustering especially when the dimensionality of data is relatively large.

7 Conclusion

We proposed a multi-task log-density gradient estimator in order to improve the existing estimator in higher-dimensional cases. Our fundamental idea is to exploit the relatedness inhering in the partial derivatives through regularized multi-task learning. As a result, we experimentally confirmed that our method significantly improves the accuracy of log-density gradient estimation. Finally, we demonstrated its usefulness of the proposed log-density gradient estimator in mode-seeking clustering.

Although fixing the similarity parameters $\gamma_{j,j'}$ to be 1 worked reasonably well in our experiments, carefully tuning them may further improve the estimation accuracy. A good practice may be to use the heuristic procedure given in Algorithm 1, whose properties have yet to be analyzed. Another way is to use Bayesian optimization techniques such as the Gaussian process approaches studied in Bergstra et al. (2013) and Snoek et al. (2012), which have been experimentally shown to be reasonably fast even in large-scale hyper-parameter tuning tasks.

As log-density gradient is a vector-valued function learning problem, one may consider applying kernel-based methods for such problems (Micchelli et al., 2005; Caponnetto et al., 2008). Micchelli et al. (2005) showed a representer theorem for a wide class of optimizations in a reproducing kernel Hilbert space of vector-valued functions whose objective functional depends only on outputs of the function to be optimized. However, it may not be possible to directly employ those methods in the LSLDG framework since the LSLDG objective functional also depends on the gradients besides outputs of the function. It is an important open question whether LSLDG also admits a similar representer theorem or not.

Log-density gradient estimation would be useful in a measure for non-Gaussianity (Huber, 1985) and other further fundamental statistical topics (Singh, 1977). In the future work, we will investigate the performance of our proposed method in these topics.

Acknowledgments

IY was supported by KAKENHI 26280054, HS was supported by KAKENHI 15H06103, and MS was supported by KAKENHI 25700022.

References

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Bandera, J. M. Pérez-Lorenzo, J. P. Bandera, and F. Sandoval. Mean shift based clustering of hough domain for fast line segment detection. *Pattern Recognition Letters*, 27(6):578–586, 2006.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

R. Beran. Adaptive estimates for autoregressive processes. *Annals of the Institute of Statistical Mathematics*, 28(1):77–89, 1976.

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2013.

S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th International Conference on Machine Learning*, pages 56–63. ACM, 2008.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

J. Catlett. Inductive learning from subsets or disposal of excess training data considered harmful. In *Australian Workshop on Knowledge Acqusition for Knowledge-Based Systems, Pokolbin*, pages 53–67, 1991.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

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.

D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2000*, volume 2, pages 142–149, 2000.

A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying. (2008). Universal multi-task kernels. *The Journal of Machine Learning Research*, 9:1615–1646, 2008.

D. D. Cox. A penalty method for nonparametric estimation of the logarithmic derivative of a density function. *Annals of the Institute of Statistical Mathematics*, 37(1):271–288, 1985.

T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117. ACM, 2004.

T. Evgeniou, C. A Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *The Journal of Machine Learning Research*, 6:615–637, 2005.

J. Fan, Q. Yao, and H. Tong. (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, 83(1):189–206, 1996.

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.

H. Hachiya, M. Sugiyama, and N. Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012.

P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

H. Kaya, P. Tüfekci, S. F. Gürgen. (2012) Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine. In *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012*, pages 13–18, Dubai, 2012. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

M. Lichman. UCI Machine Learning Repository, 2013. Irvine, CA: University of California, School of Information and Computer Science. URL `http://archive.ics.uci.edu/ml`.

D. D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, and Y. Zhang. (2013) Failure analysis of parameter-induced simulation crashes in climate models. *Geoscientific Model Development*, 6(4):1157–1171, 2013.

C. A. Micchelli and M. Pontil. On Learning Vector-valued Functions. *Neural Computation*, 17(1):177–204, 2005.

C. A. Micchelli and M. Pontil. (2004). Kernels for Multi-task Learning. In *Advances in Neural Information Processing Systems*, pages 921–928, 2004.

S. Mukherjee and D. X. Zhou. (2006). Learning coordinate covariances via gradients. *The Journal of Machine Learning Research*, 7:519–549, 2006.

G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.

H. Sasaki, A. Hyvärinen, and M. Sugiyama. Clustering via mode seeking by direct estimation of the gradient of a log-density. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2014)*, pages 19–34, Nancy, France, 2014.

R. S. Singh. Applications of estimators of a density and its derivatives to certain statistical problems. *Journal of the Royal Statistical Society. Series B*, 39(3):357–363, 1977.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

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.

S. Thrun. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

P. Tüfekci. (2014) Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*, pages 281–287, 1997.

X. Wang, C. Zhang, and Z. Zhang. Boosted multi-task learning for face verification with applications to web image and video search. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2009*, pages 142–149, 2009.

J. Warga. Minimizing certain convex functions. *Journal of the Society for Industrial & Applied Mathematics*, 11(3):588–593, 1963.

Y. Zhang. Heterogeneous-neighborhood-based multi-task local learning algorithms. In *Advances in Neural Information Processing Systems*, pages 1896–1904, 2013.