Direct Conditional Probability Density Estimation with Sparse Feature Selection

Motoki Shiga
Gifu University, Japan.
shiga_m@gifu-u.ac.jp

Voot Tangkaratt
Tokyo Institute of Technology, Japan.
voot@sg.cs.titech.ac.jp

Masashi Sugiyama
The University of Tokyo, Japan.
sugi@k.u-tokyo.ac.jp
http://www.ms.k.u-tokyo.ac.jp

Abstract

Regression is a fundamental problem in statistical data analysis, which aims at estimating the conditional mean of output given input. However, regression is not informative enough if the conditional probability density is multi-modal, asymmetric, and heteroscedastic. To overcome this limitation, various estimators of conditional densities themselves have been developed, and a kernel-based approach called *least-squares conditional density estimation* (LS-CDE) was demonstrated to be promising. However, LS-CDE still suffers from large estimation error if input contains many irrelevant features. In this paper, we therefore propose an extension of LS-CDE called *sparse additive CDE* (SA-CDE), which allows automatic feature selection in CDE. SA-CDE applies kernel LS-CDE to each input feature in an additive manner and penalizes the whole solution by a group-sparse regularizer. We also give a subgradient-based optimization method for SA-CDE training that scales well to high-dimensional large data sets. Through experiments with benchmark and humanoid robot transition datasets, we demonstrate the usefulness of SA-CDE in noisy CDE problems.

Keywords

Conditional density estimation, Feature selection, Sparse structured norm.

1 Introduction

Estimating the statistical dependency between input $\boldsymbol{x}$ and output $\boldsymbol{y}$ plays a crucial role in various real-world applications. For example, in robot transition estimation which is highly useful in *model-based reinforcement learning* (Sutton and Barto, 1998), input $\boldsymbol{x}$ corresponds to the pair of the current state of a robot and an action the robot takes, and output $\boldsymbol{y}$ corresponds to the destination state after taking the action. Another application is disease diagnosis, in which input $\boldsymbol{x}$ corresponds to measurements of biomarkers and/or clinical images and output $\boldsymbol{y}$ corresponds to the presence (or the progression level) of a disease. Thus, accurately estimating the statistical dependency is an important and fundamental problem in statistical data analysis. The most basic approach to this problem is regression, which estimates the conditional *mean* of output $\boldsymbol{y}$ given input $\boldsymbol{x}$. Regression gives the optimal estimation of output $\boldsymbol{y}$ for additive Gaussian output noise. However, if the conditional probability density of output $\boldsymbol{y}$ given input $\boldsymbol{x}$, denoted by $p(\boldsymbol{y}|\boldsymbol{x})$, possesses more complex structure such as multi-modality, asymmetry, and heteroscedasticity, estimating the conditional mean by regression is not necessarily informative.

To overcome the limitation of regression, estimation of conditional densities from paired samples $\{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$ has been investigated. The most naive approach to estimating $p(\boldsymbol{y}|\boldsymbol{x} = \widetilde{\boldsymbol{x}})$, the conditional density of output $\boldsymbol{y}$ at test input point $\boldsymbol{x} = \widetilde{\boldsymbol{x}}$, is to use the *kernel density estimator* (KDE) (Silverman, 1986) with samples such that $\|\boldsymbol{x}^{(n)} - \widetilde{\boldsymbol{x}}\|_2^2 \leq \epsilon$. However, this naive method does not work well in high-dimensional problems. Slightly more sophisticated variants have been proposed that use weighted KDE (Fan et al, 1996; Wolff et al, 1999), but they still share the same weakness.

The *mixture density network* (MDN) (Bishop, 2006) uses a mixture of parametric densities for modeling the conditional density, and the parameters are estimated by a neural network as functions of input $\boldsymbol{x}$. MDN was demonstrated to work well, but its training is time-consuming and only a local optimal solution may be found due to the non-convexity of neural network training. A similar method based on a mixture of Gaussian processes was developed (Tresp, 2001), which can be trained in a computationally more efficient way by the expectation-maximization algorithm (Dempster et al, 1977). However, due to the non-convexity of the optimization problem, it is difficult to find the global optimal solution.

Kernel quantile regression (KQR) (Takeuchi et al, 2006; Li et al, 2007) gives non-parametric percentile estimates of conditional distributions through convex optimization. KQR can be used for estimating the entire conditional cumulative distribution by solving KQR for all percentiles. It was shown that the regularization path tracking technique (Hastie et al, 2004) can be employed for efficiently computing the entire conditional cumulative distribution (Takeuchi et al, 2009). However, KQR is applicable only to one-dimensional output, which limits the range of applications significantly.

Least-squares conditional density estimation (LS-CDE) allows estimation of multiple-input-multiple-output conditional densities by directly learning a conditional density model with least-squares estimation (Sugiyama et al, 2010). For linear-in-parameter models such as a linear combination of Gaussian kernels, LS-CDE is formulated as a convex optimization problem and its solution can be obtained efficiently and analytically just by solving a system of linear equations. Furthermore, kernel LS-CDE was proved to achieve the optimal non-parametric convergence rate to the true conditional density in the minimax sense, meaning that no method can be better than LS-CDE asymptotically. Through extensive experiments, LS-CDE was demonstrated to compare favorably with competing approaches.

However, LS-CDE still suffers from large estimation error when many irrelevant features exist in input $\boldsymbol{x}$. Such irrelevant features are conceivable in many real-world problems. For example, in gene expression analysis for diseased cells, only a small subset of biomarker genes (input) affects the disease status (output). A standard way to cope with high input dimensionality is to select relevant features with forward selection or backward elimination (Guyon and Elisseeff, 2003), but this often leads to a local optimal set of features.

In this paper, we propose extending LS-CDE to allow simultaneous feature selection during conditional density estimation. More specifically, we apply kernel LS-CDE to each input feature in an additive manner and penalize the whole solution by a group-sparse regularizer (Yuan and Lin, 2006). Our subgradient-based optimization solver allows computationally efficient selection of relevant features that are even non-linearly correlated with output $\boldsymbol{y}$. Numerical experiments on noisy conditional density estimation demonstrate that our proposed method, which we call *sparse additive CDE* (SA-CDE), compares favorably with baseline approaches in estimation accuracy and computational efficiency.

The remainder of this paper is structured as follows. In Section 2, we formulate the problem of conditional density estimation and describe our proposed SA-CDE method. We experimentally evaluate the performance of SA-CDE in Section 3, and we summarize our contribution in Section 4.

2 Conditional Density Estimation with Sparse Feature Selection

In this section, we formulate the problem of conditional density estimation and describe our proposed SA-CDE method.

2.1 Problem Formulation

Let

$$\boldsymbol{x} = (x_1, \ldots, x_{D_x})^{\mathrm{T}} \in \mathbb{R}^{D_x}$$

be an input vector and $\boldsymbol{y} \in \mathbb{R}^{D_y}$ be an output vector, where $\mathbb{R}$ is the set of all real numbers, D_x is the dimension of the input vector, and D_y is the dimension of the output vector. We are given i.i.d. input-output paired samples of size N following the joint probability distribution with density $p(\boldsymbol{x}, \boldsymbol{y})$:

$$(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)}). \tag{1}$$

We assume that only some of the input features are relevant to output $\boldsymbol{y}$. Then, such relevant features are sufficient for predicting output $\boldsymbol{y}$ (Li, 1991; Cook and Ni, 2005):

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{x}^{+}), \tag{2}$$

where $\boldsymbol{x}^{+}$ is the sub-vector of $\boldsymbol{x}$ that consists only of relevant features.

Our goal is to estimate the conditional density $p(\boldsymbol{y}|\boldsymbol{x})$ from the training samples (1) via sufficient feature selection (2).

2.2 Sparse Additive Conditional Density Estimation

We use an additive conditional density model:

$$\hat{p}(\boldsymbol{y}|\boldsymbol{x}) := \sum_{d=1}^{D_x} \hat{r}_d(\boldsymbol{y}, x_d), \tag{3}$$

where $\hat{r}_d(\boldsymbol{y}, x_d)$ is an unnormalized estimator with the d-th input feature x_d. We use a linear combination of B basis functions as $\hat{r}_d(\boldsymbol{y}, x)$:

$$\begin{aligned}\hat{r}_d(\boldsymbol{y}, x) &:= \sum_{b=1}^{B} \alpha_{d,b}\Big\{\eta_b(\boldsymbol{y}) \cdot \varphi_{d,b}(x)\Big\} \\ &= \boldsymbol{\alpha}_d^{\mathrm{T}}\Big\{\boldsymbol{\eta}(\boldsymbol{y}) \circ \boldsymbol{\varphi}_d(x)\Big\}, \quad d = 1, \ldots, D_x,\end{aligned} \tag{4}$$

where

$$\boldsymbol{\alpha}_d = (\alpha_{d,1}, \ldots, \alpha_{d,B})^{\mathrm{T}} \in \mathbb{R}_{+}^{B}$$

is a parameter vector for the d-th input x_d,

$$\boldsymbol{\eta}(\boldsymbol{y}) = (\eta_1(\boldsymbol{y}), \ldots, \eta_B(\boldsymbol{y}))^{\mathrm{T}}$$

is a vector of basis functions for output $\boldsymbol{y}$,

$$\boldsymbol{\varphi}_d(x) = (\varphi_{d,1}(x), \ldots, \varphi_{d,B}(x))^{\mathrm{T}}$$

is a vector of basis functions for the d-th input x_d, and $\circ$ denotes the Hadamard (or element-wise) product. We use the following Gaussian kernels as the basis functions:

$$\eta_b(\boldsymbol{y}) := \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{\nu}_b\|^2}{2\sigma^2}\right), \quad b = 1, \ldots, B, \tag{5}$$

$$\varphi_{d,b}(x_d) := \exp\left(-\frac{(x_d - \mu_{d,b})^2}{2\sigma^2}\right), \quad d = 1, \ldots, D_x, \quad b = 1, \ldots, B, \tag{6}$$

where $\boldsymbol{\nu}_b$ and $\mu_{d,b}$ are the Gaussian centers and σ is the Gaussian width. The Gaussian centers are fixed at the points chosen randomly from training samples. On the other hand, the Gaussian width will be optimized by cross-validation (see Section 2.5). For simplicity, we assume that input $\boldsymbol{x}$ and output $\boldsymbol{y}$ of the training samples (1) are both normalized element-wise to have the unit variance in advance and use the common Gaussian width σ. Note that the magnitude of parameter $\boldsymbol{\alpha}_d$ may reflect the relevance between the d-th feature and output $\boldsymbol{y}$.

Figures 1 and 2 illustrate two examples of the additive function model (3). The first example shown in Figure 1 is

$$r_1^{(A)}(y, x_1) = \mathcal{N}(y|\sin(x_1), \frac{1}{4}) \quad \text{and} \quad r_2^{(A)}(y, x_2) = 0,$$

while the second example shown in Figure 2 is

$$r_1^{(B)}(y, x_1) = \frac{1}{2}\mathcal{N}(y|\sin(x_1), \frac{1}{4}) \quad \text{and} \quad r_2^{(B)}(y, x_2) = \frac{1}{2}\mathcal{N}(y|\frac{1}{4}x_2, \frac{1}{4}).$$

Note that $r_2(y, x_2)$ is only different in these two examples. In the density function $p^{(A)}(y|x_1, x_2)$ shown in Figure 1(c), x_2 does not affect output y. On the other hand, in density function $p^{(B)}(y|x_1, x_2)$ shown in 2(c), output y varies depending on x_2. This illustrates that the magnitude of $r_d^{(B)}(y, x_d)$, which is controlled by $\boldsymbol{\alpha}_d$, may reflect the relevance between the d-th feature and output $\boldsymbol{y}$.

Our optimization parameters to be learned in model (3) are

$$\boldsymbol{\alpha} := (\boldsymbol{\alpha}_1^{\mathrm{T}}, \ldots, \boldsymbol{\alpha}_{D_x}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{B \cdot D_x}.$$

We learn $\boldsymbol{\alpha}$ to minimize the squared error between true conditional density $p(\boldsymbol{y}|\boldsymbol{x})$ and our estimator $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$

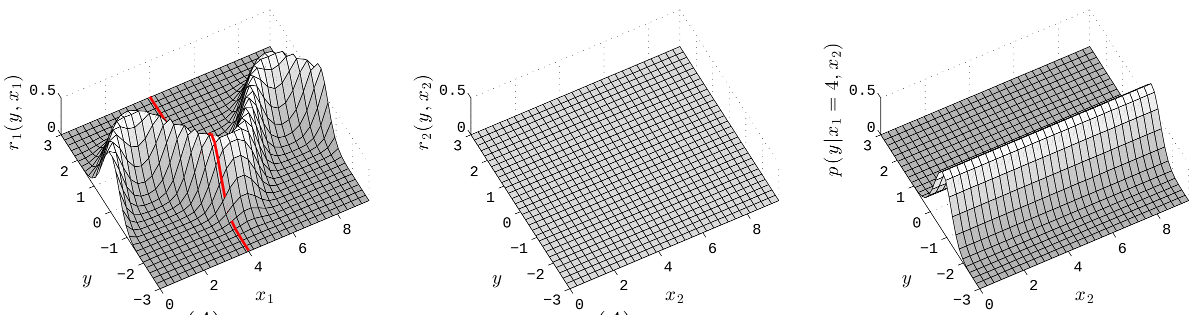
$$J_0(\boldsymbol{\alpha}) := \frac{1}{2}\int\int\Big(\hat{p}(\boldsymbol{y}|\boldsymbol{x}) - p(\boldsymbol{y}|\boldsymbol{x})\Big)^2 p(\boldsymbol{x})d\boldsymbol{y}d\boldsymbol{x}. \tag{7}$$

Substituting our model (3) and $p(\boldsymbol{y}, \boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$ into (7), we have

$$\begin{aligned} J_0(\boldsymbol{\alpha}) &= \frac{1}{2}\sum_{d_1,d_2=1}^{D_x}\int\int\int \hat{r}_{d_1}(\boldsymbol{y}, x_{d_1})\hat{r}_{d_2}(\boldsymbol{y}, x_{d_2})p(x_{d_1}, x_{d_2})d\boldsymbol{y}dx_{d_1}dx_{d_2} \\ &\quad -\sum_{d=1}^{D_x}\int\int \hat{r}_d(\boldsymbol{y}, x_d)p(\boldsymbol{y}, x_d)d\boldsymbol{y}dx_d + \quad \text{Const.} \end{aligned} \tag{8}$$

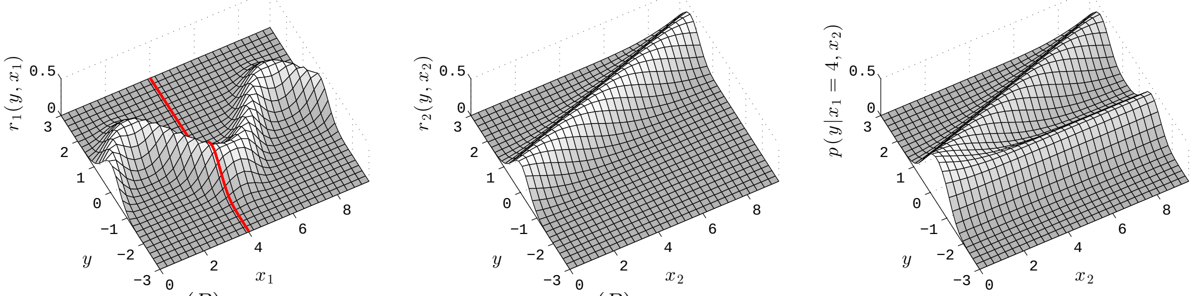
Using empirical approximation and ignoring the constant term, we can approximate the loss function as follows (the detailed derivation is described in Appendix A):

$$\begin{aligned} \hat{J}_0(\boldsymbol{\alpha}) &= \frac{1}{2N}\sum_{d_1,d_2=1}^{D_x}\sum_{n=1}^{N}\int \hat{r}_{d_1}\Big(\boldsymbol{y}, x_{d_1}^{(n)}\Big)\hat{r}_{d_2}\Big(\boldsymbol{y}, x_{d_2}^{(n)}\Big)d\boldsymbol{y} \\ &\quad -\frac{1}{N}\sum_{d=1}^{D_x}\sum_{n=1}^{N}\hat{r}_d\Big(\boldsymbol{y}^{(n)}, x_d^{(n)}\Big) \qquad (9) \\ &= \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\alpha} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{\alpha}, \qquad (10) \end{aligned}$$



(a) $r_1^{(A)}(y, x_1)$ (b) $r_2^{(A)}(y, x_2)$ (c) $p^{(A)}(y|x_1 = 4, x_2)$

Figure 1: Conditional density function in which output y is relevant to only input x_1.



(a) $r_1^{(B)}(y, x_1)$ (b) $r_2^{(B)}(y, x_2)$ (c) $p^{(B)}(y|x_1 = 4, x_2)$

Figure 2: Conditional density function in which output y is relevant to both input x_1 and x_2.

where $\boldsymbol{H} \in \mathbb{R}^{B\cdot D_x \times B\cdot D_x}$ and $\boldsymbol{h} \in \mathbb{R}^{B\cdot D_x}$ are given by

$$\boldsymbol{H} := \begin{pmatrix} \boldsymbol{H}_{1,1} & \cdots & \boldsymbol{H}_{1,D_x} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H}_{D_x,1} & \cdots & \boldsymbol{H}_{D_x,D_x} \end{pmatrix}. \tag{11}$$

$$\begin{aligned}[\boldsymbol{H}_{d_1,d_2}]_{b_1,b_2} := {} & \frac{\left(\sqrt{\pi}\sigma\right)^{D_y}}{N} \exp\left(-\frac{\|\boldsymbol{\nu}_{b_1} - \boldsymbol{\nu}_{b_2}\|^2}{4\sigma^2}\right) \\ & \times \sum_{n=1}^{N} \exp\left(-\frac{(x_{d_1}^{(n)} - \mu_{d_1,b_1})^2}{2\sigma^2} - \frac{(x_{d_2}^{(n)} - \mu_{d_2,b_2})^2}{2\sigma^2}\right), \\ & d_1, d_2 = 1, \ldots, D_x,\ b_1, b_2 = 1, \ldots, B, \end{aligned} \tag{12}$$

$$\boldsymbol{h} := \left(\boldsymbol{h}_1^{\mathrm{T}}, \ldots, \boldsymbol{h}_{D_x}^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{13}$$

$$\begin{aligned} h_{d,b} := {} & \frac{1}{N} \sum_{n=1}^{N} \exp\left(-\frac{\|\boldsymbol{y}^{(n)} - \boldsymbol{\nu}_b\|^2}{2\sigma^2} - \frac{(x_d^{(n)} - \mu_{d,b})^2}{2\sigma^2}\right), \\ & d = 1, \ldots, D_x,\ b = 1, \ldots, B. \end{aligned} \tag{14}$$

To perform feature selection in our additive CDE model, we introduce an (ℓ_1, ℓ_2)-mixed norm:

$$\Omega(\boldsymbol{\alpha}) := \sum_{d=1}^{D_x} \left\| \boldsymbol{\alpha}_d \right\|_2. \tag{15}$$

In the mixed norm $\Omega(\boldsymbol{\alpha})$, the parameter vector $\boldsymbol{\alpha}$ is grouped in the sub-vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{D_x}$. Minimizing a loss function penalized with $\Omega(\boldsymbol{\alpha})$ tends to produce a group-wise sparse solution (Yuan and Lin, 2006), which means that the penalized optimization can be useful for selecting a relevant subset of input variables:

$$\min_{\boldsymbol{\alpha} \geq 0} J(\boldsymbol{\alpha}) := \hat{J}_0(\boldsymbol{\alpha}) + \lambda \, \Omega(\boldsymbol{\alpha}), \tag{16}$$

where $\lambda \geq 0$ is the regularization parameter. Note that empirical squared error $\hat{J}_0(\boldsymbol{\alpha})$ is differentiable and convex because the Hessian matrix $\boldsymbol{H}$ is positive definite. $\Omega(\boldsymbol{\alpha})$ is also convex but non-differentiable. Overall, (16) is a convex optimization problem and we develop a fast optimization algorithm below.

2.3 Optimization Algorithm

We use a *proximal method* (Sra et al, 2012; Beck and Teboulle, 2009) to solve the optimization problem (16). More specifically, we consider a linear approximation to function $\hat{J}_0$ at the current solution $\boldsymbol{\alpha}^{(t)}$, penalized by a proximal term to keep the update confined in the neighborhood:

$$\min_{\boldsymbol{\alpha} \geq 0} \left\{ \hat{J}_0\left(\boldsymbol{\alpha}^{(t)}\right) + \nabla \hat{J}_0\left(\boldsymbol{\alpha}^{(t)}\right)^{\mathrm{T}} \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^{(t)}\right) + \lambda \cdot \Omega(\boldsymbol{\alpha}) + \frac{L}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\alpha}^{(t)} \right\|_2^2 \right\}. \tag{17}$$

Here, L is the Lipchitz constant[1], which is given by the maximum eigenvalue of $\boldsymbol{H}$ in the current setup. We can describe our update rule analytically as follows (the detailed derivation is described in Appendix B):

$$\boldsymbol{\alpha}_d^{(t+1)} \leftarrow \left[1 - \frac{\lambda}{L \cdot \|\boldsymbol{u}_d\|_2} \right]_+ \cdot \boldsymbol{u}_d, \quad d = 1, \ldots, D_x, \tag{18}$$

where $\boldsymbol{u}_d$ is the sub-vector of $\boldsymbol{u}$ associated with the d-th feature:

$$\boldsymbol{u} := \left[\boldsymbol{\alpha}^{(t)} - \frac{1}{L} \nabla \hat{J}_0\left(\boldsymbol{\alpha}^{(t)}\right) \right]_+. \tag{19}$$

The operator $[\cdot]_+$ rounds up negative values to zero. This update rule follows the standard proximal method, meaning that it has the global convergence rate of $O\left(\frac{1}{t}\right)$, where t is the number of update iterations.

[1]The Lipchitz constant L for $f(x)$ satisfies $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{y}\|_2$ for arbitrary $\boldsymbol{x}$ and $\boldsymbol{y}$.

Algorithm 1 SA-CDE

```
1: Initialization: $t \leftarrow 0$, $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$, and $\epsilon > 0$.
2: Compute $\boldsymbol{H}$ and $\boldsymbol{h}$ by (11) and (13), respectively.
3: Set $L$ to the maximum eigenvalue of $\boldsymbol{H}$.
4: **repeat**
5:     $\boldsymbol{u} \leftarrow \left[\boldsymbol{\alpha}^{(t)} - \frac{1}{L}\left(\boldsymbol{H}\boldsymbol{\alpha}^{(t)} - \boldsymbol{h}\right)\right]_+$
6:     $\boldsymbol{\alpha}_d^{(t+1)} \leftarrow \left[1 - \frac{\lambda}{L \cdot \|\boldsymbol{u}_d\|_2}\right]_+ \cdot \boldsymbol{u}_d$,   for $d = 1$ to $D_x$.
7: **until** $\|\vec{\alpha}^{(t+1)} - \vec{\alpha}^{(t)}\| < \epsilon$
8: Normalize the optimized conditional density by (20).
```

2.4 Post Processing

Because we did not explicitly include the normalization constraint, the optimized conditional density estimator $\hat{r}(\boldsymbol{y}, \boldsymbol{x}) = \sum_{d=1}^{D_x} \hat{r}_d(\boldsymbol{y}, x_d)$ may not be integrated to one with respect to $\boldsymbol{y}$. Here, we renormalize the estimator after optimization as

$$\hat{p}(\boldsymbol{y}|\boldsymbol{x}) \quad = \quad \frac{\hat{r}(\boldsymbol{y}, \boldsymbol{x})}{\int \hat{r}(\boldsymbol{y}', \boldsymbol{x}) d\boldsymbol{y}'}. \tag{20}$$

The denominator can be analytically calculated as

$$\int \hat{r}(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{y} = \left(\sqrt{2\pi}\sigma\right)^{D_y} \sum_{d=1}^{D_x} \boldsymbol{\alpha}_d^{\mathrm{T}} \boldsymbol{\varphi}_d\big(x_d, \boldsymbol{\mu}_d\big). \tag{21}$$

Algorithm 2.4 summarizes our algorithm, which we call *sparse additive CDE* (SA-CDE).

2.5 Cross-Validation for Model Selection

Performance of SA-CDE depends on the choice of model parameters such as the Gaussian width σ and the regularization parameter λ. Cross-validation (CV) is available to systematically choose these model parameters. Throughout this paper, we use 5-fold CV: we first divide the samples into five subsets, then learn the parameter using four subsets, and evaluate the test error using the held-out subset. This procedure is iterated five times with different training-test choice and the error is averaged.

We use the negative log-likelihood (NLL) as our metric for evaluating the test error:

$$\mathrm{NLL} = -\frac{1}{|\mathcal{T}|} \sum_{n \in \mathcal{T}} \log \hat{p}(\boldsymbol{y}^{(n)} | \boldsymbol{x}^{(n)}), \tag{22}$$

where $\mathcal{T}$ is the set of indices of test samples. The smaller the value of NLL is, the better the performance of the conditional density estimator is. Thus, we chose the model parameters that minimize the averaged NLL by 5-fold CV.

3 Numerical Experiments

In this section, we experimentally evaluate the performance of our proposed method, SA-CDE. Throughout the experiments, the number of basis functions is fixed to $B = \min(100, N)$. The model parameters σ and λ were chosen from the twenty values between 10^{-2} and 2 at the equal interval in the logarithmic scale by 5-fold CV. We use NLL (22) for the performance measure of conditional density estimation. NLL is computed from test samples, which are not used for learning parameters and hyper-parameters. All experiments were implemented by Matlab 2013b and an HP DL360p Gen8 E5 v2 server with two CPUs of Xeon E5-2650 v2 2.60GHz (8 Core) and the main memory of 96 GB.

3.1 Compared Methods

We compare SA-CDE with the following methods:

- **Sparse additive feature selection LSCDE (SA-LSCDE):** SA-LSCDE is a variation of the proposed SA-CDE, which first runs SA-CDE for feature selection and then estimates the conditionally density by LS-CDE with only selected features.

- **ϵ-neighbor kernel density estimation (eKDE):** eKDE estimates a conditional density by standard kernel density estimation using neighborhood samples in the domain of input $\boldsymbol{x}$, denoted by $\mathcal{I}_{\boldsymbol{x},\epsilon} := \{\boldsymbol{x}^{(i)} : \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|_2^2 \le \epsilon\}$ for threshold ϵ. In the case of Gaussian kernels, eKDE is given as

$$\hat{p}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{|\mathcal{I}_{\boldsymbol{x},\epsilon}|} \sum_{i \in \mathcal{I}_{\boldsymbol{x},\epsilon}} \mathcal{N}(\boldsymbol{y}, \boldsymbol{y}^{(i)}, \sigma^2 \boldsymbol{I}_{D_y}), \tag{23}$$

  where $\mathcal{N}(\boldsymbol{y}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian density function with respect to $\boldsymbol{y}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and $\boldsymbol{I}_{D_x}$ is the identity matrix of size D_x. In experiments, threshold ϵ and bandwidth σ were chosen based on 5-fold CV with respect to NLL, where the candidate values of ϵ are the twenty values between 10^{-2} and 5 at the equal interval in the logarithmic scale.

- **Least-squares conditional density estimation (LS-CDE):** The original LS-CDE method. This corresponds to a multi-dimensional non-sparse version of SA-CDE where, instead of the group-sparse penalty and an additive model, an ℓ_2-penalty $\lambda\|\boldsymbol{\alpha}\|_2^2$ and a multi-dimensional linear-in-parameter model,

$$\begin{aligned}\hat{p}(\boldsymbol{y}|\boldsymbol{x}) &:= \hat{r}(\boldsymbol{y}, \boldsymbol{x}) \\ &= \sum_{b=1}^{B} \alpha_b \Big\{ \eta_b(\boldsymbol{y}) \cdot \varphi_b(\boldsymbol{x}) \Big\},\end{aligned} \tag{24}$$

  is used. We use the Gaussian kernels for both $\eta_b(\cdot)$ and $\varphi_b(\cdot)$, where the bandwidth σ and the regularization parameter λ are chosen based on 5-fold CV with respect to NLL.

Table 1: Computational complexities of our methods and existing CDEs.

Method	SA-CDE	SA-LSCDE	LS-CDE	e-KDE	NW-CDE
Time	$O(N^3D_x^3)$	$O(N^3D_x^3)$	$O(N^3)$	$O(N^2D_x)$	$O(N^2D_x)$
Space	$O(N^2D_x^2)$	$O(N^2D_x^2)$	$O(N^2)$	$O(N^2)$	$O(N^2)$

Table 2: Computational complexities of our methods and existing CDEs with forward feature selection

Method	SA-CDE	SA-LSCDE	FW-LSCDE	FW-eKDE	FW-NWCDE
Time	$O(N^3D_x^3)$	$O(N^3D_x^3)$	$O(D_x!N^3)$	$O(D_x!N^2)$	$O(D_x!N^2)$
Space	$O(N^2D_x^2)$	$O(N^2D_x^2)$	$O(N^2)$	$O(N^2)$	$O(N^2)$

- **Nadaraya-Watson CDE (NW-CDE):** This corresponds to a simple version of LS-CDE, which fixes weights of basis functions to $\frac{1}{B}$:

$$\begin{aligned}\hat{p}(\boldsymbol{y}|\boldsymbol{x}) &:= \frac{\hat{p}(\boldsymbol{y},\boldsymbol{x})}{\hat{p}(\boldsymbol{x})} \\ &= \frac{\sum_{b=1}^{B}\eta_b(\boldsymbol{y})\cdot\varphi_b(\boldsymbol{x})}{\sum_{b=1}^{B}\varphi_b(\boldsymbol{x})}. \end{aligned} \tag{25}$$

  We use the Gaussian kernels for both $\eta_b(\cdot)$ and $\varphi_b(\cdot)$, where the bandwidth σ is chosen based on leave-one-out CV for the exact likelihood formulated in Holmes et al (2007). To directly employ the method in Holmes et al (2007), we only use B samples in this CV procedure.

- **Forward feature selection + eKDE (FW-eKDE):** Forward feature selection is performed based on 5-fold CV with respect to NLL. That is, the most useful feature that maximally reduces the cross-validated NLL by eKDE is selected one by one until the cross-validated NLL no longer decreases.

- **Forward feature selection + LS-CDE (FW-LSCDE):** Similarly, forward feature selection is performed for LS-CDE.

- **Forward feature selection + NW-CDE (FW-NWCDE):** Similarly, forward feature selection is performed for NW-CDE.

Tables 1 and 2 summarize the time and space complexities of our proposed method (SA-CDE and SA-LSCDE) and compared methods for a fixed hyper-parameter, under the assumption of $N \gg D_x \gg D_y$ and $O(N) = O(B)$. All time complexities for T candidates of hyper-parameters are T times larger than those in Tables 1 and 2, while all space complexities does not depend on T. The time complexities of e-KDE and NW-CDE are the smallest, while CDEs with forward feature selections require larger computational costs.

Especially, FW-LSCDE is worst in terms of the computational complexity. Thanks to the single-shot procedure of feature selection, SA-CDE and SA-LSCDE are computationally much more efficient than FW-LSCDE. The space complexities of all methods except for SA-CDE and SA-LSCDE are $O(N^2)$, while those of SA-CDE and SA-LSCDE are $O(N^2 D_x^2)$, which increases with D_x. Overall, the time complexities of SA-CDE and SA-LSCDE are much smaller than CDEs with forward selection procedures in return for increasing the space complexity.

3.2 Illustrative Examples

We first illustrate the behavior of our proposed method, SA-CDE, using toy and benchmark datasets having one relevant feature x_1 and five irrelevant features $x_2, \ldots, x_6$,

- **Toy data 1:** x_1 is independently generated following the uniform distribution on $[-1, 1]$, while each of $x_2, \ldots, x_6$ is generated by $x_1 + \epsilon_c$ where ϵ_c is a noise variable following the normal distribution with mean 0 and standard deviation $3\hat{\sigma}$, and $\hat{\sigma}$ is the standard deviation of x_1. Output y is generated as a function of x_1 as

$$y|x_1 \quad \sim \quad \mathrm{sinc}\left(\frac{3}{4}\pi x_1\right) + \frac{1}{8}\exp\big(1 - x_1\big) \cdot \varepsilon, \tag{26}$$

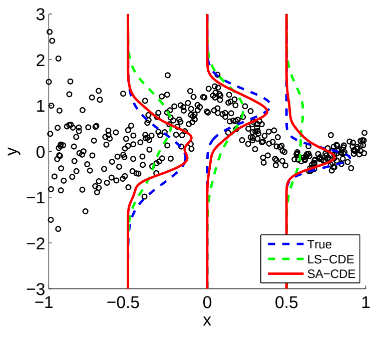
  where ε is standard normal noise. We generate $N = 300$ samples for estimating the conditional density.

- **Old Faithful Geyser:** A benchmark dataset with $D_x = D_y = 1$ that consists of durations of $N = 299$ eruptions of the Old Faithful Geyser (Weisberg, 1985). We add five irrelevant features $x_2, \ldots, x_6$ in a similar manner to Toy data 1.

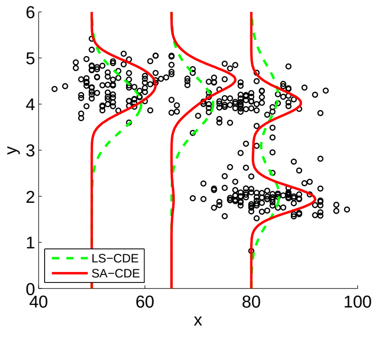
- **Bone Mineral Density:** A benchmark dataset with $D_x = D_y = 1$ that consists of relative spinal bone mineral density measurements on $N = 485$ North American adolescents (Hastie et al, 2001). We add five irrelevant features $x_2, \ldots, x_6$ in a similar manner to Toy data 1.

Figures 3 and 4 show estimation results for these three datasets. Figure 3 shows estimated conditional densities: the black circles denote training samples, and the red solid and green dashed lines denote the estimates obtained by the proposed SA-CDE and the plain LS-CDE, respectively. Figure 3(a) also contains the true conditional density drawn by the blue dashed line. Figure 4 shows the regularization paths of SA-CDE, *i.e.* the magnitude of each learned parameter $\|\boldsymbol{\alpha}_d\|_2$ as a function of the regularization parameter (Hastie et al, 2004). The blue line denotes the path of the relevant feature, x_1, while lines with other colors denote the paths of irrelevant features $x_2, \ldots, x_6$. The vertical black dashed line indicates the value of λ selected by 5-fold CV.

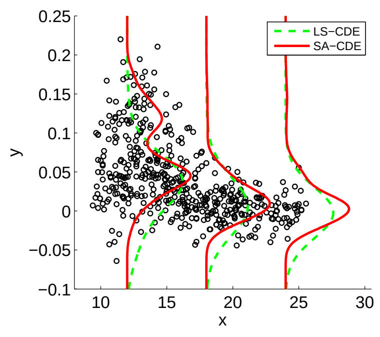
The regularization paths in Figure 4 show that, in (a) Toy data 1 and (b) Old Faithful Geyser, the parameters corresponding to the irrelevant feature are zero and that corresponding to the relevant feature is non-zero for the cross-validated solution, which means



(a) Toy data 1

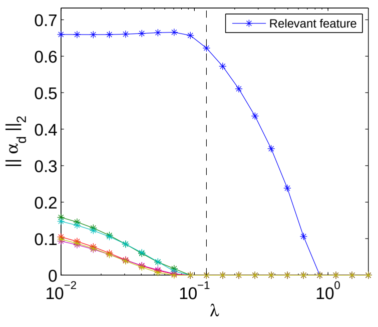


(b) Old Faithful Geyser

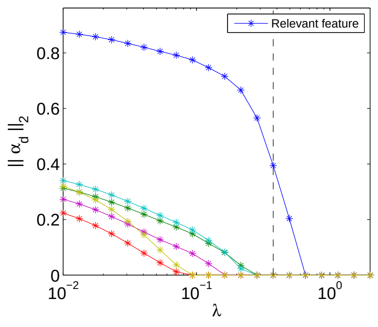


(c) Bone Mineral Density

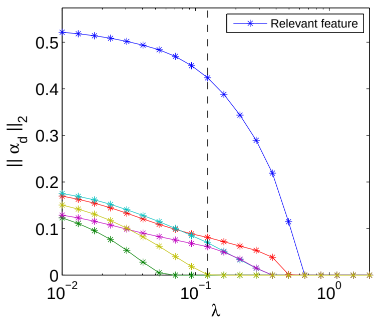
Figure 3: Training samples and estimated conditional densities.



(a) Toy data 1



(b) Old Faithful Geyser



(c) Bone Mineral Density

Figure 4: Regularized path of SA-CDE. The blue line denotes the path of the relevant feature, x_1, while lines with other colors denote paths of irrelevant features $x_2, \ldots, x_6$. The vertical black dashed line indicates the value of λ selected by 5-fold CV.

that SA-CDE optimally performs feature selection. In Figure 4(c) Bone Mineral Density, some of irrelevant features are non-zero because features with the skewed distribution are strongly correlated with the relevant feature despite additive Gaussian noise. Thus these features may still contain some information on the output value.

The estimation results in Figure 3(a) show that, SA-CDE gives more accurate estimates than the plain LS-CDE. In Figures 3(b) and 3(c), SA-CDE tends to give sharper conditional density estimates than the plain LS-CDE. This is because relatively large Gaussian kernel widths are chosen in LS-CDE to incorporate irrelevant noisy features. This indicates that LS-CDE with many irrelevant features tend to produce too flat conditional densities which are not informative, while SA-CDE can avoid this problem by automatically eliminating irrelevant features.

3.3 Comparison of Performance and Computation Time for Different Numbers of Samples

We compare the estimation performance and computation time of our proposed methods with existing CDEs for different numbers of training samples, $N = 200, 500, 1000, 2500, 5000$. Performance evaluation measures are NLL and MSE (Mean Squared Error) in (7), both of which are computed from 10^4 test samples. The integral in MSE (7) is computed based on the test sample average. Datasets in this experiments are

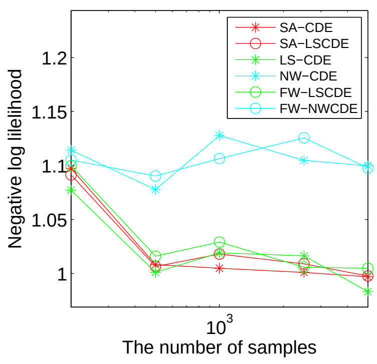
- **Toy data 1:** The generation procedure is the same as the one in the previous section, in which both dimensions of relevant feature (input) and output are one.
- **Toy data 2:** Each irrelevant $x_1, x_2, x_4, x_5, \ldots, x_{D_x-2}, x_{D_x-1}$ is independently generated following the uniform distribution on $[-1, 1]$. Relevant features are generated by $x_{3d} = x_{3d-2} + x_{3d-1}$, and the d-th dimension of output $\boldsymbol{y}$ is generated as a function of $x_{3d},\ d = 1, 2, \ldots, D_x/3$ as

$$y_d | x_{3d} \quad \sim \quad \mathrm{sinc}\left(\frac{3}{4}\pi x_{3d}\right) + \frac{1}{8}\exp\left(1 - x_{3d}\right) \cdot \varepsilon, \tag{27}$$

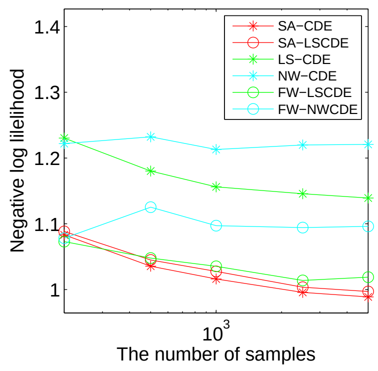
  where ε is standard normal noise. This dataset has multi-dimensional relevant features and outputs.

Figures 5 and 6 show NLL and MSE on Toy data 1, and Figures 7 and 8 NLL and MSE on Toy data 2. These figures show that CDEs with feature selection except for FW-NWCDE, *i.e.* SA-CDE, SA-LSCDE, and FW-LSCDE, decrease both NLL and MSE with increasing the number of samples. However FW-NWCDE cannot decrease estimation errors even when the number of samples is increased. This is because NW-CDE does not optimize weight values of basis functions, resulting in poor performance. For Toy data 1 which has only single relevant input and output, the performance of SA-CDE and SA-LSCDE is almost the same. However, for Toy data 2 which has multiple relevant inputs and outputs, SA-LSCDE is much better than SA-CDE. This is because SA-LSCDE can represent more complex conditional densities than SA-CDE which is limited to additive models.

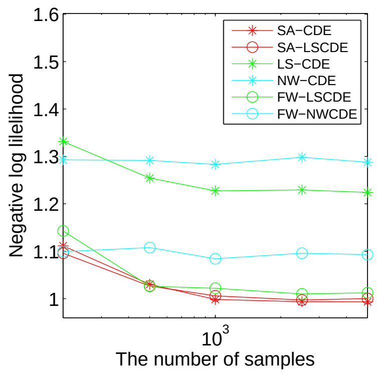
The computation time on Toy data 1 and Toy data 2, which is actual run-time including both 5-fold CV to optimize hyper-parameters and conditional density estimation for test data, are shown in Figures 9 and 10, respectively. These figures show that the computation time of FW-LSCDE is much longer than others when the number of input features is larger than 2 because of the time consuming procedure of forward feature selection. This weakness becomes more critical when the number of training samples is increasing. These results show that our proposed methods, SA-CDE and SA-LSCDE, are much faster than the existing feature selection methods, and our methods tend to improve the performance by feature selection.



(a) $D_x = 2$

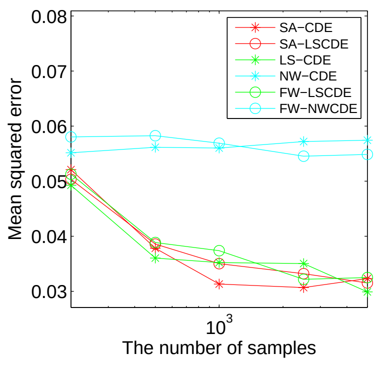


(b) $D_x = 3$

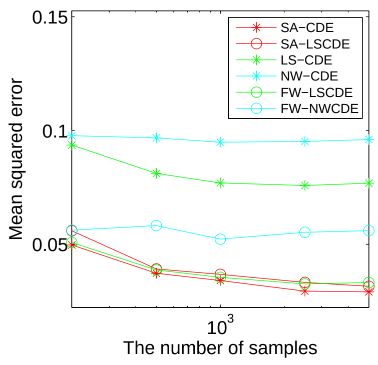


(c) $D_x = 4$

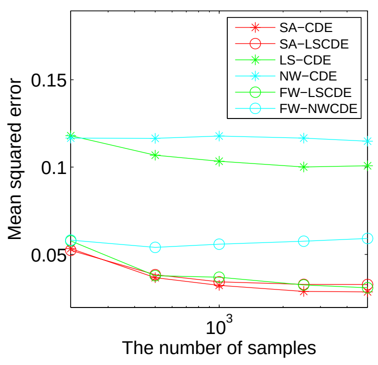
Figure 5: Negative log likelihood on Toy data 1 ($D_y = 1$).



(a) $D_x = 2$



(b) $D_x = 3$

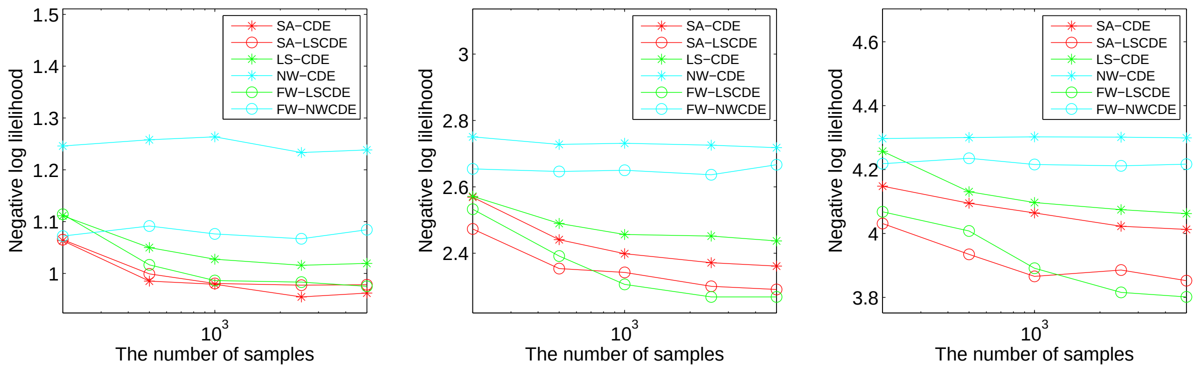


(c) $D_x = 4$

Figure 6: Mean squared error on Toy data 1 ($D_y = 1$).

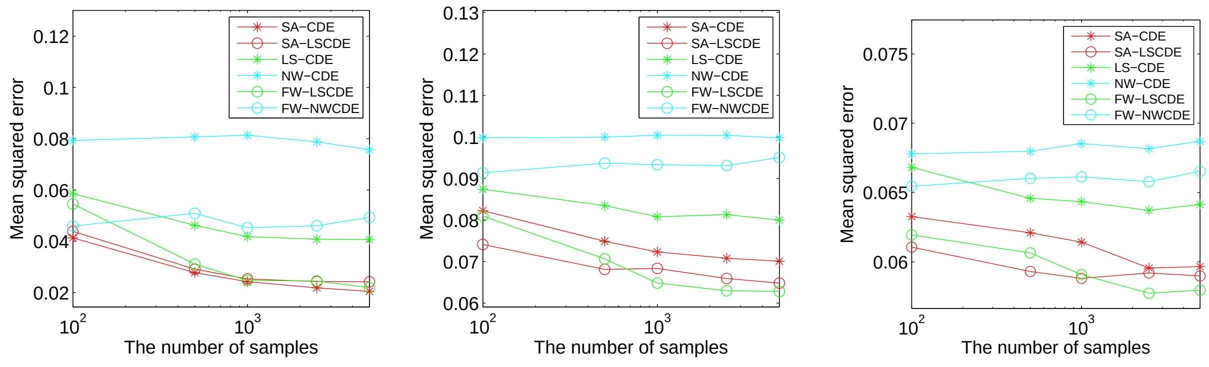
3.4 Hyper-Parameter Selection

Our proposed methods (SA-CDE and SA-LSCDE) need to optimize hyper-parameters based on CV, which is a time consuming procedure. Thus it is a bottleneck of their computational time. Here, we compare the performance and computation time when the number of hyper-parameter candidates is changed. We keep choosing the hyper-parameters σ and λ between 10^{-2} and 2 at the equal interval in the logarithmic scale, and we only change the number of hyper-parameter candidates: $n_g = 5, 10, 20$. $n_g = 20$ is the default setting for all other experiments. Datasets and evaluation measures we use in this experiments are the same as Section 3.3. These experimental results are shown in Figures 11–16. All performance results in Figure 11–14 show that SA-CDE and SA-LSCDE with $n_g = 20$ are the best. On the other hand, those with $n_g = 5$, which cannot improve the performance even when the number of samples increases, are the worst. On the other hand, in terms of the computation time, Figures 15 and 16 show that the $n_g = 5$ is the best, while $n_g = 20$ is the worst. This demonstrates the tradeoff between the performance and computation time on optimizing hyper-parameters using CV. We note



(a) $D_x = 3, D_y = 1$ (b) $D_x = 6, D_y = 2$ (c) $D_x = 9, D_y = 3$

Figure 7: Negative log likelihood on Toy data 2.



(a) $D_x = 3, D_y = 1$ (b) $D_x = 6, D_y = 2$ (c) $D_x = 9, D_y = 3$

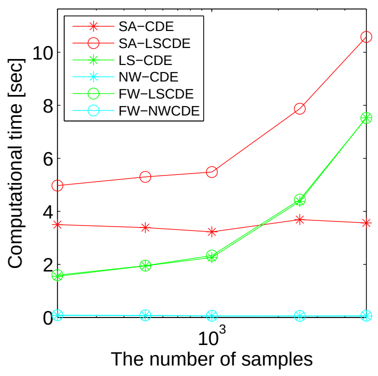
Figure 8: Mean squared error on Toy data 2.

that the computation time of SA-CDE and SA-LSCDE with $n_g = 20$ is the worst, but they are still much faster than FW-LSCDE. Thus we keep using $n_g = 20$ in all numerical experiments of later sections.

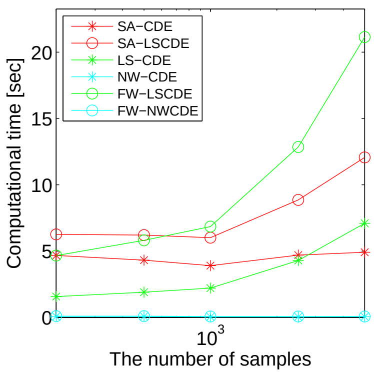
3.5 Performance Comparison for Different Numbers of Irrelevant Features

Next, we compare the performance of SA-CDE with LS-CDE, eKDE, FW-LSCDE, and FW-eKDE for different numbers of irrelevant features. We use three datasets: (a) the same Toy data 1 ($N = 300$ and $D_x = 1$) used in the previous experiments, (b) the same *Old Faithful Geyser* benchmark dataset ($N = 299$ and $D_x = 1$), and (c) the *crabs* benchmark dataset ($N = 200$ and $D_x = 6$) taken from the R package[2]. For each dataset, we add m $(= 0, 1, \ldots, 10)$ irrelevant features by copying x_1 and adding Gaussian noise in a similar manner to the previous experiments. We randomly choose a half of samples

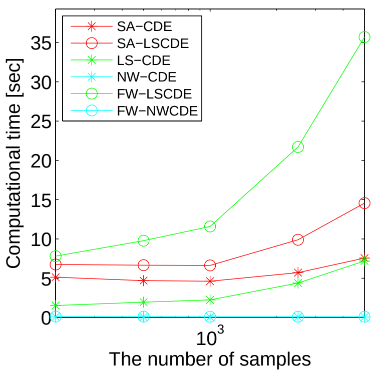
[2]`http://www.r-project.org/`.



(a) $D_x = 2$

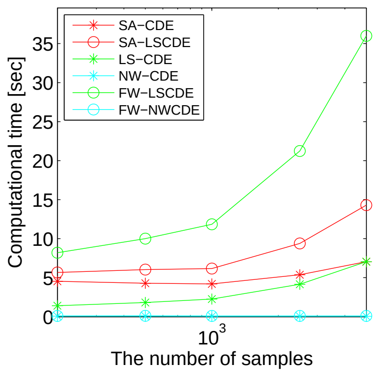


(b) $D_x = 3$

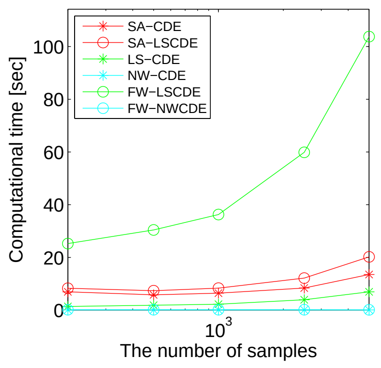


(c) $D_x = 4$

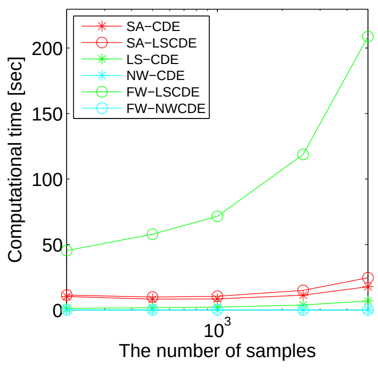
Figure 9: Computational time on Toy data 1 ($D_y = 1$).



(a) $D_x = 3, D_y = 1$



(b) $D_x = 6, D_y = 2$



(c) $D_x = 9, D_y = 3$

Figure 10: Computational time on Toy data 2.

as training samples to estimate conditional densities and use the rest as test samples to compute the test NLL. This procedure is repeated 100 times and the averaged test NLL is computed. The experimental results are summarized in Figure 17.

In all three cases, the NLL values of LS-CDE, NW-CDE, and eKDE (without feature selection) grow as the number of irrelevant features increases. On the other hand, the NLL values of SA-CDE, SA-LSCDE, FW-LSCDE, FW-NWCDE, and FW-eKDE do not grow that much when the number of irrelevant features increases. This clearly demonstrates an advantage of performing feature selection.

The total computation time of each method, *i.e.* run-time including 5-fold CV to optimize hyper-parameters and conditional density estimation for test data, is plotted in Figure 18. This shows that all methods require more computation costs as the number of features increases. The computation time of FW-eKDE, FW-LSCDE, and FW-NWCDE is much longer than the plain eKDE, LS-CDE, and NW-CDE, implying that forward feature selection is highly time-consuming. Indeed, forward feature selection involves repetitious conditional density estimation to find the best feature to add, which is com-

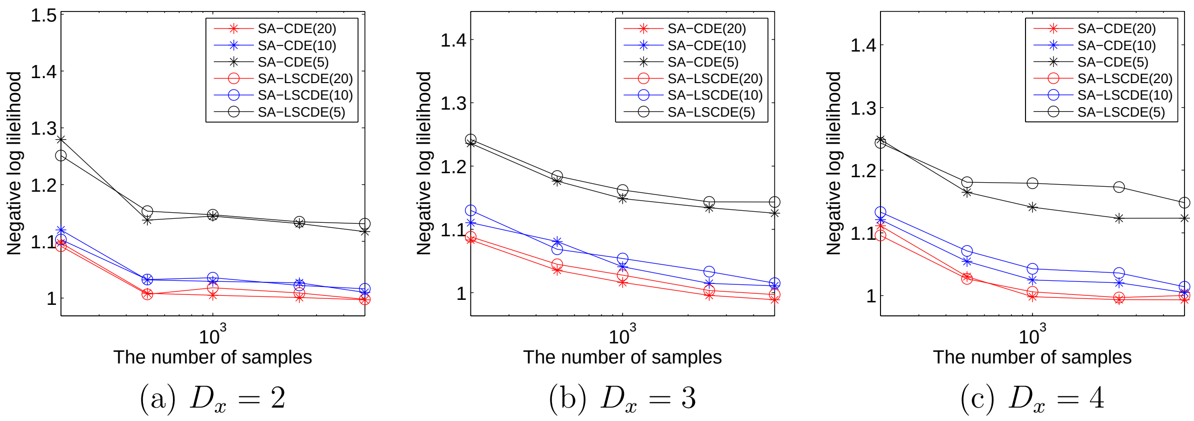


Figure 11: Negative log likelihood on Toy data 1 ($D_y = 1$).

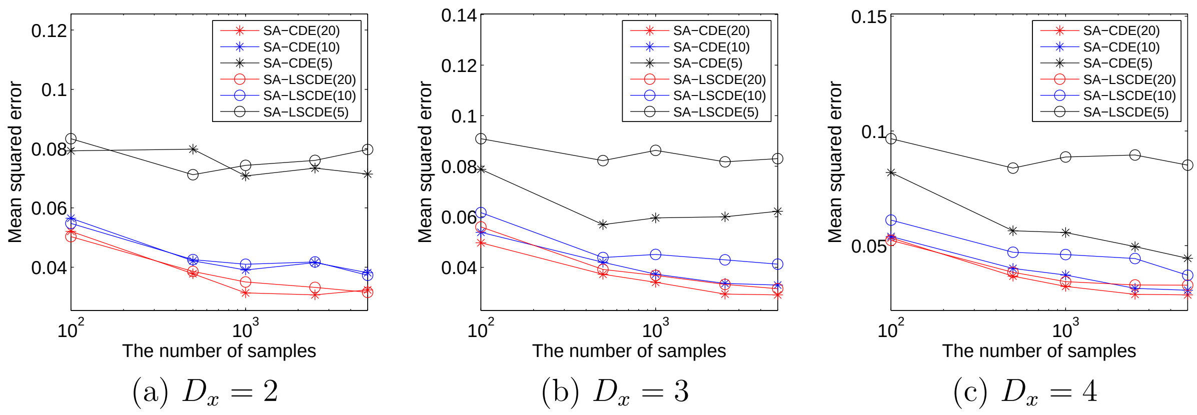
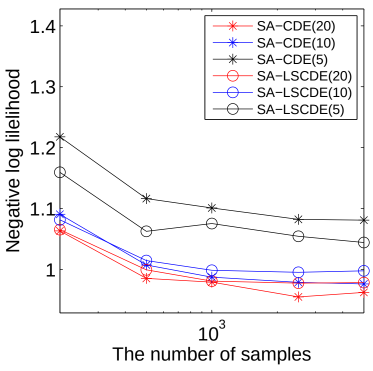


Figure 12: Mean squared error on Toy data 1 ($D_y = 1$).

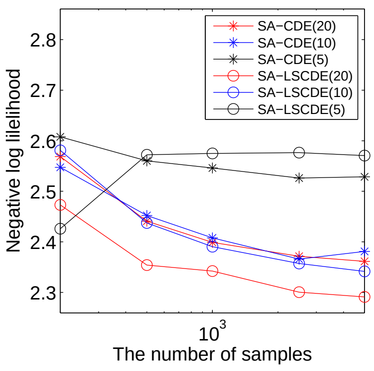
putationally highly demanding. On the other hand, the computation time of SA-CDE and SA-LSCDE grows less sharply than FW-LSCDE, thanks to the single-shot procedure of feature selection and conditional density estimation.

3.6 Benchmark Datasets

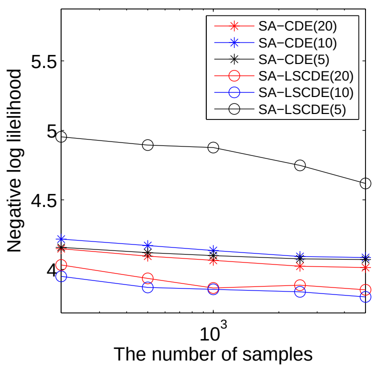
We further compare the performance of the proposed SA-CDE with other methods on twelve benchmark datasets accompanied with the R package. All of these datasets have one-dimensional output, *i.e.*, $D_y = 1$. The number of features $|\mathcal{F}|$ and the number of samples N are listed in Table 3. For all datasets, five irrelevant features are added which are the copy of the relevant variable or a linear combination of two relevant variables contaminated with Gaussian noise. The type of noise and relevant variables are chosen at random. Gaussian noise is generated in a similar manner to the previous experiments. In this experiment, we randomly choose a half of samples as training data for estimating conditional densities, and the rest is used as test data for computing the test NLL. The experimental results are summarized in Table 3. The values described in the table are



(a) $D_x = 3, D_y = 1$

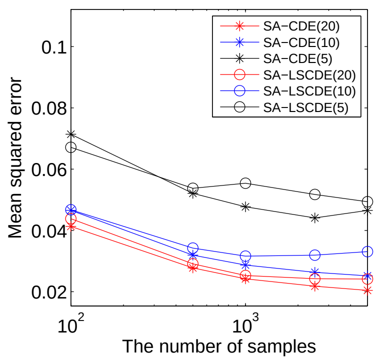


(b) $D_x = 6, D_y = 2$

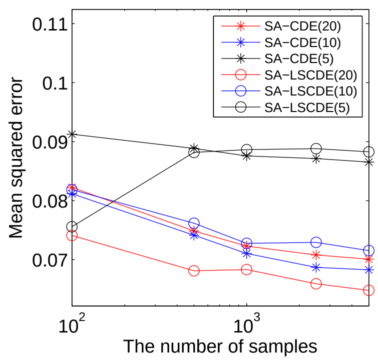


(c) $D_x = 9, D_y = 3$

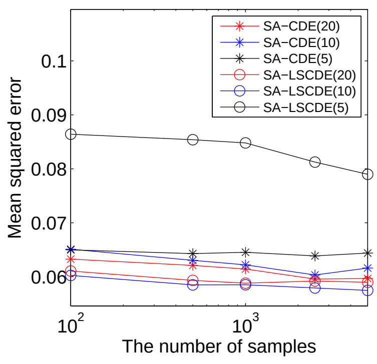
Figure 13: Negative log likelihood on Toy data 2.



(a) $D_x = 3, D_y = 1$



(b) $D_x = 6, D_y = 2$

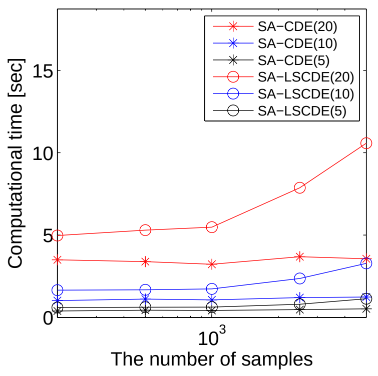


(c) $D_x = 9, D_y = 3$

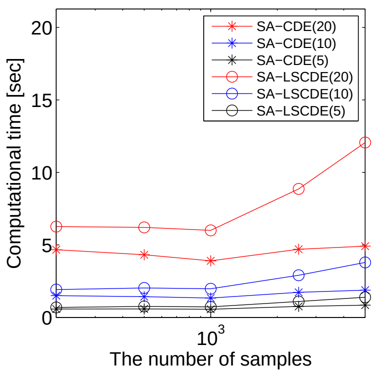
Figure 14: Mean squared error on Toy data 2.

averaged NLL values and standard deviations over one hundred runs with different random seeds. The bold letter means the best NLL and comparable results that could not be rejected by the two-sided paired t-test at significance level 5%. The bottom row shows the averaged normalized computation time.

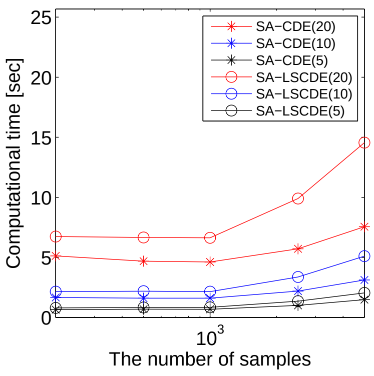
Table 3 shows that the performance of our methods (SA-CDE and SA-LSCDE) is best on nine datasets. For high-dimensional datasets, especially when $|\mathcal{F}|$ is seven or more, SA-CDE tends to outperform other methods with statistical significance. For low-dimensional datasets with large N, the performance of SA-LSCDE outperforms SA-CDE because of their expressive power of functions. For low-dimensional datasets with small N, FW-NWCDE performs the best because all other methods optimizing weights of basis functions cause overfitting. LSCDE, eKDE, NW-CDE, and FW-eKDE are computationally much more efficient than SA-CDE and SA-LSCDE, but these methods tend to perform poorly for high-dimensional relevant features with noisy dimensions.



(a) $D_x = 2$

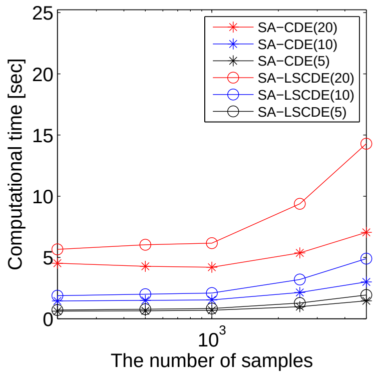


(b) $D_x = 3$

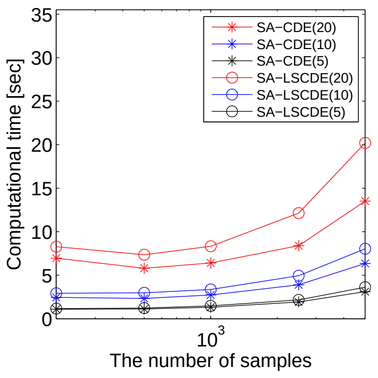


(c) $D_x = 4$

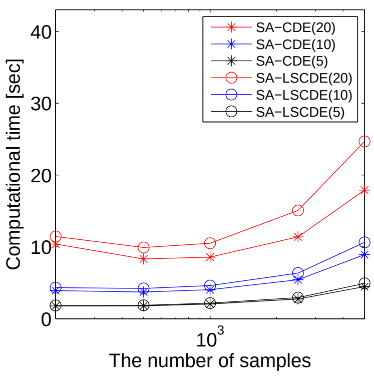
Figure 15: Computational time on Toy data 1 ($D_y = 1$).



(a) $D_x = 3, D_y = 1$



(b) $D_x = 6, D_y = 2$



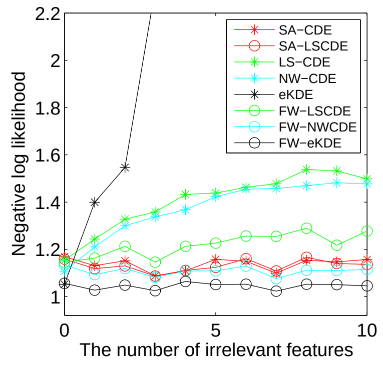
(c) $D_x = 9, D_y = 3$

Figure 16: Computational time on Toy data 2.

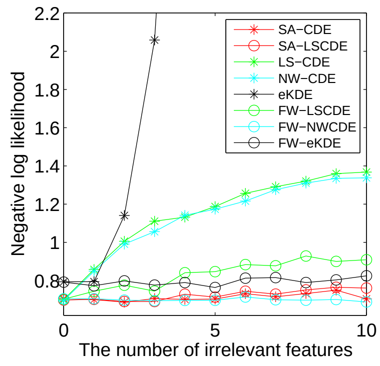
3.7 Humanoid Robot Transition Dataset

Finally, we evaluate the performance of the proposed method on humanoid robot transition estimation with multiple inputs and multiple outputs. The dataset was generated from a simulator of the upper-body part of the humanoid robot *CB-i* (Cheng et al, 2007). The robot has 9 controllable joints: shoulder pitch, shoulder roll and elbow pitch of the right arm, shoulder pitch, shoulder roll and elbow pitch of the left arm, wait yaw, torso roll, and torso pitch joints.

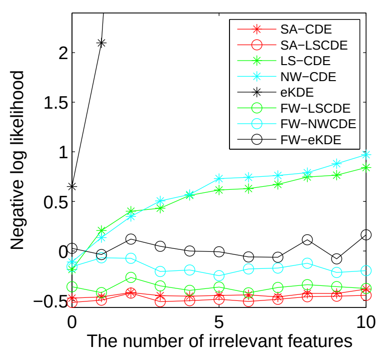
Posture of the robot is described by 18-dimensional real-valued state vector $\boldsymbol{s}$, which corresponds to the angle and angular velocity of each joint in radians and radians per seconds, respectively. We can control the robot by sending the action command $\boldsymbol{a}$ to the system. The action command $\boldsymbol{a}$ is a 9-dimensional real-valued vector, which corresponds to the target angle of each joint. When the robot is currently at state $\boldsymbol{s}$ and receive action $\boldsymbol{a}$, the physical control system of the simulator calculates the amount of torques to be applied to each joint. These torques are calculated by the *Proportional-Derivative* (PD)



(a) Toy data 1

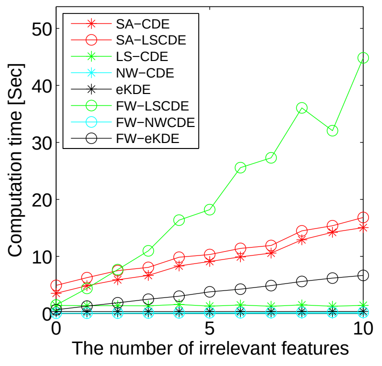


(b) Old Faithful Geyser

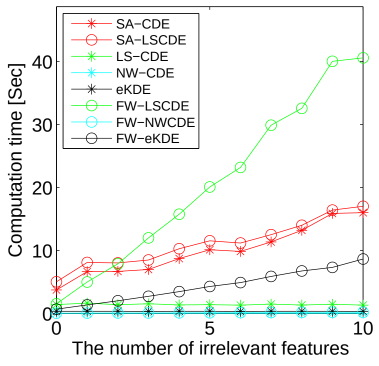


(c) Crabs

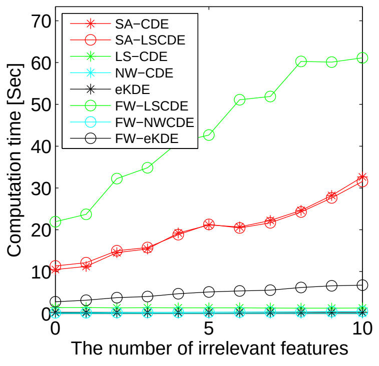
Figure 17: Negative log likelihood with increasing noise dimensions



(a) Toy data 1



(b) Old Faithful Geyser



(c) Crabs

Figure 18: Computation time

controller as

$$\tau_i = K_{p_i}(a_i - s_i) - K_{d_i}\dot{s}_i, \tag{28}$$

where s_i, $\dot{s}_i$, and a_i denote the current angle, the current angular velocity, and the received target angle of the i-th joint, respectively. K_{p_i} and K_{d_i} denote the position and velocity gains for the i-th joint, respectively. We set $K_{p_i} = 200$ and $K_{d_i} = 10$ for the elbow pitch joints and $K_{p_i} = 2000$ and $K_{d_i} = 100$ for the other joints. After the torques are applied to the joints, the physical system updates the state of the robot to $\boldsymbol{s}'$. We simulate a noisy control system by perturbing action vectors with independent bi-modal Gaussian noise. More specifically, for each action element, we add Gaussian noise with mean 0 and standard deviation 0.052 with probability 0.6, and Gaussian noise with mean -0.087 and standard deviation 0.052 with probability 0.4.

To generate transition samples, we first generated the initial posture of the robot $\boldsymbol{s}^{(1)}$ at random and then simulated a trajectory with 100 steps, *i.e.* $\boldsymbol{s}^{(2)}, \ldots, \boldsymbol{s}^{(100)}$. For each step, we additionally generated m irrelevant input features $\boldsymbol{z}^{(n)} \in \mathbb{R}^m$ by copying a relevant variable or by linearly combining two relevant variables contaminated with

Table 3: NLL for benchmark datasets with five dimensional irrelevant features.

Name	$\|\mathcal{F}\|$	N	SA-CDE	SA-LSCDE	LS-CDE	eKDE	NW-CDE
caution	2	100	1.34 ± 0.6	$\mathbf{1.24 \pm 0.4}$	1.38 ± 0.3	24.25 ± 3.4	1.36 ± 0.2
CobarOre	2	38	1.71 ± 0.5	1.70 ± 0.4	$\mathbf{1.62 \pm 0.2}$	31.81 ± 2.9	$\mathbf{1.62 \pm 0.2}$
snowgeese	2	45	$\mathbf{1.80 \pm 2.0}$	$\mathbf{1.76 \pm 1.8}$	1.85 ± 1.3	22.04 ± 6.1	$\mathbf{1.59 \pm 1.0}$
topo	2	52	1.17 ± 0.3	1.14 ± 0.3	1.22 ± 0.2	29.30 ± 3.0	1.21 ± 0.1
sniffer	4	125	0.70 ± 0.6	$\mathbf{0.60 \pm 0.7}$	0.85 ± 0.2	16.91 ± 3.2	0.83 ± 0.2
crabs	6	200	-0.44 ± 0.1	$\mathbf{-0.47 \pm 0.3}$	0.53 ± 0.1	26.03 ± 3.1	0.58 ± 0.1
UN3	6	125	$\mathbf{1.27 \pm 0.2}$	$\mathbf{1.35 \pm 0.4}$	1.57 ± 0.6	33.36 ± 1.6	1.54 ± 0.6
birthwt	7	189	$\mathbf{1.49 \pm 0.2}$	$\mathbf{1.52 \pm 0.1}$	$\mathbf{1.51 \pm 0.1}$	31.77 ± 1.6	1.67 ± 0.2
cpus	7	209	$\mathbf{0.36 \pm 0.6}$	0.80 ± 0.7	1.19 ± 0.5	22.29 ± 3.4	1.17 ± 0.6
gilgais	8	365	$\mathbf{0.70 \pm 0.2}$	0.89 ± 0.2	1.16 ± 0.2	27.77 ± 2.2	1.11 ± 0.2
BigMac	9	69	1.33 ± 0.8	1.37 ± 0.7	1.42 ± 0.7	35.79 ± 0.5	1.34 ± 0.5
highway	11	39	$\mathbf{1.38 \pm 0.7}$	1.60 ± 0.7	1.71 ± 0.8	36.04 ± 0.0	1.74 ± 0.7
Time			1.00	0.00	0.06	0.02	0.00

(Continuation of Table 3)

Name	FW-LSCDE	FW-eKDE	FW-NWCDE
caution	1.33 ± 0.6	1.35 ± 0.6	1.30 ± 0.5
CobarOre	1.95 ± 0.6	2.45 ± 1.9	$\mathbf{1.65 \pm 0.4}$
snowgeese	2.09 ± 1.9	3.03 ± 2.4	$\mathbf{1.82 \pm 1.8}$
topo	1.19 ± 0.4	1.73 ± 1.2	$\mathbf{1.07 \pm 0.2}$
sniffer	0.74 ± 0.8	0.96 ± 0.8	0.96 ± 1.1
crabs	-0.37 ± 0.3	0.08 ± 0.6	-0.12 ± 0.8
UN3	$\mathbf{1.27 \pm 0.3}$	1.60 ± 0.6	$\mathbf{1.34 \pm 0.3}$
birthwt	1.67 ± 0.2	1.75 ± 0.5	3.85 ± 2.1
cpus	0.70 ± 0.8	1.00 ± 0.9	0.76 ± 0.9
gilgais	0.76 ± 0.2	0.97 ± 0.3	1.20 ± 0.3
BigMac	1.45 ± 0.9	2.54 ± 1.7	$\mathbf{1.23 \pm 0.8}$
highway	2.06 ± 1.0	3.17 ± 1.9	2.18 ± 1.8
Time	2.73	0.54	0.01

Gaussian noise in a similar manner to the previous experiments. By iterating these procedures, we obtained the transition samples $\{(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{s}'^{(n)})\}_{n=1}^{10000}$.

Our goal is to learn the system dynamics as state transition probability $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ from these samples. Thus, as the conditional density estimation problem, the state-action pair $(\boldsymbol{s}^{\mathrm{T}}, \boldsymbol{a}^{\mathrm{T}}, \boldsymbol{z}^{\mathrm{T}})^{\mathrm{T}}$ is regarded as input variable $\boldsymbol{x}$, while the next state $\boldsymbol{s}'$ is regarded as output variable $\boldsymbol{y}$. Note that an accurate estimate of the state transition probability is highly useful in *model-based reinforcement learning* (Sutton and Barto, 1998).

From the transition samples, we randomly picked up 5000 samples as training data and used the other 5000 samples as test data to calculate NLL. We compare our proposed method SA-CDE with LS-CDE, NW-CDE, FW-LSCDE, and FW-NWCDE, as well as parametric conditional density estimation by the *Gaussian process regression* (GP-CDE) (Rasmussen and Williams, 2005). In this experiment, the candidate values of regularization parameter λ are the twenty values between 10^{-3} and 10^{-1} at the equal interval in the logarithmic scale, while the candidate values of other parameters are the same as the previous setting. We consider three datasets with $J = 2, 4, 9$ joints, and change the number of irrelevant features as $m = 0, 5, 10, 15, 20$. Thus, the input dimensionality is $3J + m$, while the output dimensionality is $2J$. For each J and m, we evaluated the performance of conditional density estimation methods by averaged NLL and averaged computational time over 20 runs.

Figure 19 shows the experimental results, where the solid lines denote the averaged values and the dashed lines denote the averaged values with one standard deviation. From the plots, we can confirm that the NLL values of LS-CDE, NW-CDE, and GP-CDE grow sharply as the number of irrelevant features m is increased. On the other hand, the NLL values of SA-CDE, FW-LSCDE, FW-NWCDE, and SA-LSCDE do not increase even if the number of irrelevant features is increased. Among them, SA-CDE cannot outperform FW-LSCDE and SA-LSCDE, because the additive-model assumption of SA-CDE caused large estimation bias. However, feature selection by SA-CDE itself performs well and SA-LSCDE performs comparably to FW-LSCDE.

Figure 20 plots the computation time. LS-CDE, NW-CDE, and GP-CDE are very fast because no feature selection process is involved. NW-CDE and FW-NWCDE are also very fast because no optimization of weight parameters is involved. Among SA-CDE, FW-LSCDE, and SA-LSCDE, SA-CDE and SA-LSCDE are much faster than FW-LSCDE.

Overall, in this challenging task of robot transition estimation, SA-LSCDE, the combination of SA-CDE and LS-CDE, was shown to be the most promising approach.

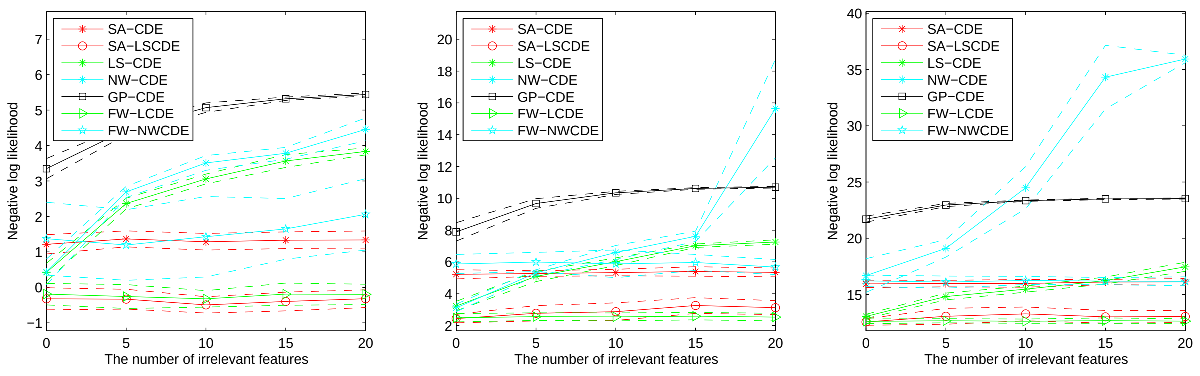
4 Conclusions

We proposed a direct estimator of conditional probability densities that is equipped with feature selection. Our feature selection strategy is based on the ℓ_1/ℓ_2 mixed-norm, which tends to produce a group-sparse solution. An optimization algorithm based on a proximal method was presented that is guaranteed to possess fast convergence. The numerical experiments on benchmark and robot transition datasets demonstrated that the proposed method is promising.

SA-CDE assumes the additive structure for feature selection. However, this causes linear increase of the time and space complexities, resulting in high computation costs for datasets with a large number of features. Improving the scalability issue is future work.

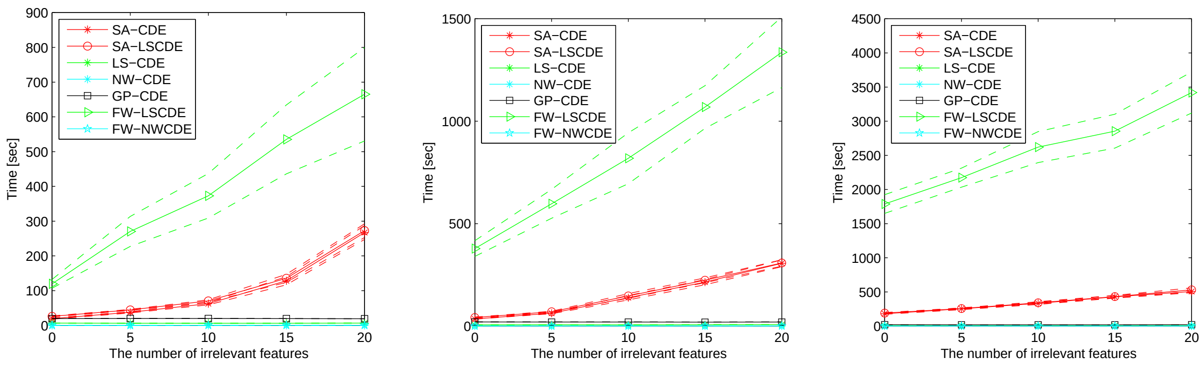
Acknowledgements

Motoki Shiga was supported by JSPS KAKENHI 25870322. Masashi Sugiyama was supported by JSPS KAKENHI 23120004 and AOARD. Authors thank Dr. Ichiro Takeuchi, Nagoya Institute of Technology, for kindly providing his source codes.



(a) $J = 2$ (b) $J = 4$ (c) $J = 9$

Figure 19: Negative log likelihood on humanoid robot data.



(a) $J = 2$ (b) $J = 4$ (b) $J = 9$

Figure 20: Computation time on humanoid robot data.

A Derivation of Equation (10)

Substituting (4) into (9), we can transform the term associated with d_1, d_2, and n in the above equation as

$$\begin{aligned}
&\int \hat{r}_{d_1}\left(\boldsymbol{y}, x_{d_1}^{(n)}\right) \hat{r}_{d_2}\left(\boldsymbol{y}, x_{d_2}^{(n)}\right) d\boldsymbol{y} \\
&= \int \prod_{d=d_1,d_2} \left\{ \sum_{b=1}^{B} \alpha_{d,b} \cdot \eta\left(\boldsymbol{y}, \boldsymbol{\nu}_b\right) \cdot \varphi_d\left(x_d^{(n)}, \mu_{d,b}\right) \right\} d\boldsymbol{y} \\
&= \sum_{b_1=1}^{B} \sum_{b_2=1}^{B} \left\{ \int \eta\left(\boldsymbol{y}, \boldsymbol{\nu}_{b_1}\right) \cdot \eta\left(\boldsymbol{y}, \boldsymbol{\nu}_{b_2}\right) d\boldsymbol{y} \ \times \ \varphi_{d_1}\left(x_{d_1}^{(n)}, \mu_{d_1,b_1}\right) \cdot \varphi_{d_2}\left(x_{d_2}^{(n)}, \mu_{d_2,b_2}\right) \right\}. \quad (29)
\end{aligned}$$

The integral with respect to $\boldsymbol{y}$ in the above equation can be analytically computed as

$$\begin{aligned}\int \eta(\boldsymbol{y}, \boldsymbol{\nu}_{b_1}) \cdot \eta(\boldsymbol{y}, \boldsymbol{\nu}_{b_2}) d\boldsymbol{y} &= \int \exp\left(-\frac{\|\boldsymbol{y}-\boldsymbol{\nu}_{b_1}\|^2 + \|\boldsymbol{y}-\boldsymbol{\nu}_{b_2}\|^2}{2\sigma^2}\right) d\boldsymbol{y} \\ &= \left(\sqrt{\pi}\sigma\right)^{D_y} \exp\left(-\frac{\|\boldsymbol{\nu}_{b_1}-\boldsymbol{\nu}_{b_2}\|_2^2}{4\sigma^2}\right). \qquad (30)\end{aligned}$$

Using (5), (6) and (30), we can derive (10).

B Derivation of Update Rule (18)

By some simple algebra, the optimization problem (17) can be transformed into

$$\min_{\boldsymbol{\alpha}\geq 0} \left\{ \frac{1}{2} \|\boldsymbol{\alpha} - \boldsymbol{u}\|_2^2 + \frac{\lambda}{L}\, \Omega(\boldsymbol{\alpha}) \right\}, \qquad (31)$$

where $\boldsymbol{u}$ is defined as (19). Because the optimization problem (31) is strongly convex, update of $\boldsymbol{\alpha}$ by (31) can be regarded as an operator that maps $\boldsymbol{u}$ to a unique value. This is called a proximal operator associated with $\frac{\lambda}{L}\Omega$. Denoting the proximal operator by $\mathrm{Prox}_{\frac{\lambda}{L}\Omega}(\cdot)$, we can write our update rule as

$$\begin{aligned}\boldsymbol{\alpha}^{(t+1)} &\leftarrow \mathrm{Prox}_{\frac{\lambda}{L}\Omega}(\boldsymbol{u}) \\ &:= \arg\min_{\boldsymbol{\alpha}\geq 0} \left\{ \frac{1}{2} \|\boldsymbol{\alpha} - \boldsymbol{u}\|_2^2 + \frac{\lambda}{L}\, \Omega(\boldsymbol{\alpha}) \right\}. \qquad (32)\end{aligned}$$

Next we describe the solution of this proximal operation by transforming the operator into the dual form. By using the Fenchel conjugate of function $\hat{J}_0$ and the dual norm of Ω, the dual problem of (31) can be obtained as

$$\max_{\boldsymbol{v}} \left\{ -\sup_{\boldsymbol{\alpha}} \left[\boldsymbol{\alpha}^{\mathrm{T}}(-\boldsymbol{v}) - \frac{1}{2}\|\boldsymbol{\alpha}-\boldsymbol{u}\|_2^2 \right] \right\} \quad \text{subject to} \quad \|\boldsymbol{v}_d\|_2 \leq \frac{\lambda}{L}, \quad d = 1, \ldots, D_x \qquad (33)$$

where $\boldsymbol{v}$ is the dual of $\boldsymbol{\alpha}$ and $\boldsymbol{v}_d$ is the sub-vector associated with the d-th feature. The objective function can be transformed into

$$-\sup_{\boldsymbol{\alpha}} \left[\boldsymbol{\alpha}^{\mathrm{T}}(-\boldsymbol{v}) - \frac{1}{2}\|\boldsymbol{\alpha}-\boldsymbol{u}\|_2^2 \right] = \inf_{\boldsymbol{\alpha}} \left[\frac{1}{2}\|\boldsymbol{\alpha}-\boldsymbol{u}\|_2^2 + \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{v} \right]. \qquad (34)$$

Since the above function is quadratic with respect to $\boldsymbol{\alpha}$, its minimum is achieved at the point where the derivative is zero: from this, we have $\boldsymbol{\alpha} = \boldsymbol{u} - \boldsymbol{v}$. Substituting this back into (34), we have

$$\begin{aligned}\inf_{\boldsymbol{\alpha}} \left[\frac{1}{2}\|\boldsymbol{\alpha}-\boldsymbol{u}\|_2^2 + \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{v} \right] &= \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}-\boldsymbol{u}\|_2^2 + (\boldsymbol{u}-\boldsymbol{v})^{\mathrm{T}}\boldsymbol{v} \\ &= -\frac{1}{2}\Big(\|\boldsymbol{v}-\boldsymbol{u}\|_2^2 - \|\boldsymbol{u}\|_2^2 \Big). \qquad (35)\end{aligned}$$

Further substituting (35) into (33), we can express the dual of the proximal operator for optimizing dual vector $\boldsymbol{v}$, denoted by $\mathrm{Proj}_{\frac{\lambda}{L}\Omega}$, as

$$\begin{aligned}\boldsymbol{v}^{(t+1)} &\leftarrow \mathrm{Proj}_{\frac{\lambda}{L}\Omega}(\boldsymbol{u}) \\ &:= \arg\min_{\boldsymbol{v}} \left[\frac{1}{2}\left\|\boldsymbol{v}-\boldsymbol{u}\right\|_2^2\right] \text{ subject to } \|\boldsymbol{v}_d\|_2 \leq \frac{\lambda}{L},\ d=1,\ldots,D_x. \end{aligned} \tag{36}$$

Given that the solution for each sub-vector $\boldsymbol{v}_d$ can be obtained separately, it is easy to confirm that the solution of the above proximal operation is given by

$$\left[\mathrm{Proj}_{\frac{\lambda}{L}\Omega}(\boldsymbol{u})\right]_d = \left(1-\left[1-\frac{\lambda}{L\|\boldsymbol{u}_d\|_2}\right]_+\right)\boldsymbol{u}_d, \quad d=1,\ldots,D_x. \tag{37}$$

Since $\boldsymbol{\alpha}=\boldsymbol{u}-\boldsymbol{v}$, we have

$$\mathrm{Prox}_{\frac{\lambda}{L}\Omega}(\boldsymbol{u}) = \boldsymbol{u} - \mathrm{Proj}_{\frac{\lambda}{L}\Omega}(\boldsymbol{u}). \tag{38}$$

Using the above equation, we can describe our update rule analytically as (18).

References

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Img Sci 2(1):183–202

Bishop CM (2006) Pattern Recognition and Machine Learning. Springer, New York, NY, USA

Cheng C, Hyon SH, Morimoto J, Ude A, Hale JG, Colvin G, Scroggin W, Jacobsen SC (2007) Cb: a humanoid research platform for exploring neuroscience. Advanced Robotics 21(10):1097–1114

Cook RD, Ni L (2005) Sufficient dimension reduction via inverse regression. Journal of the American Statistical Association 100(470):410–428

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, series B 39(1):1–38

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

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. Journal of Machine Learning Research 3(Mar.):1157–1182

Hastie T, Tibshirani R, Friedman J (2001) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, USA

Hastie T, Rosset S, Tibshirani R, Zhu J (2004) The entire regularization path for the support vector machine. Journal of Machine Learning Research 5:1391–1415

Holmes MP, Gray AG, Isbell CL (2007) Fast nonparametric conditional density estimation. In: Proc. of the Twenty-Third Conference Annual Conference on Uncertainty in Artificial Intelligence, pp 175–182

Li K (1991) Sliced inverse regression for dimension reduction. Journal of the American Statistical Association 86(414):316–342

Li Y, Liu Y, Zhu J (2007) Quantile regression in reproducing kernel Hilbert spaces. Journal of the American Statistical Association 102(477):255–268

Rasmussen CE, Williams CKI (2005) Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press

Silverman BW (1986) Density Estimation for Statistics and Data Analysis. Chapman and Hall, London, UK

Sra S, Nowozin S, Wright S (2012) Optimization for Machine Learning. Neural Information Processing Series, Mit Press

Sugiyama M, Takeuchi I, Suzuki T, Kanamori T, Hachiya H, Okanohara D (2010) Least-squares conditional density estimation. IEICE Transactions on Information and Systems E93-D(3):583–594

Sutton RS, Barto AG (1998) Introduction to Reinforcement Learning, 1st edn. MIT Press, Cambridge, MA, USA

Takeuchi I, Le QV, Sears TD, Smola AJ (2006) Nonparametric quantile estimation. Journal of Machine Learning Research 7:1231–1264

Takeuchi I, Nomura K, Kanamori T (2009) Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression. Neural Computation 21(2):533–559

Tresp V (2001) Mixtures of Gaussian processes. In: Leen TK, Dietterich TG, Tresp V (eds) Advances in Neural Information Processing Systems 13, MIT Press, Cambridge, MA, USA, pp 654–660

Weisberg S (1985) Applied Linear Regression. John Wiley, New York, NY, USA

Wolff RCL, Yao Q, Hall P (1999) Methods for estimating a conditional distribution function. Journal of the American Statistical Association 94(445):154–163

Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B 68(1):49–67