ISDE : Independence Structure Density Estimation

Louis Pujol

ISDE : Independent Structure Density Estimation

Louis Pujol*

Abstract

Density estimation appears as a subroutine in many learning procedures, so it is of crucial interest to have efficient methods to perform it in practical situations. Multidimensional density estimation faces the curse of dimensionality. To tackle this issue, a solution is to add a structural hypothesis through an undirected graphical model on the underlying distribution. We propose ISDE (Independence Structure Density Estimation) an algorithm designed to estimate a density and an undirected graphical model from a special family of graphs corresponding to an independence structure, where features can be separated into independent groups. It is designed for moderately high dimensional data (up to 15 features) and it can be used in parametric as well as nonparametric situations. Existing methods on nonparametric graph estimation focus on multidimensional dependencies only through pairwise ones. ISDE does not suffer from this restriction and can addresses structures not covered yet by available algorithms. In this paper, we present existing theory about independence structure, explain the construction of our algorithm and prove its effectiveness on simulated data both quantitatively, through measures of density estimation performance under Kullback-Leibler loss and qualitatively, in terms of recovering of independence structures. We also provide information about running time.

1 INTRODUCTION

Unsupervised Learning and Density Estimation Unsupervised learning is one of the most fundamental problem in data analysis. Its aim is to design methods in order to extract meaningful information from a dataset without any prior knowledge. A central problem in this field is density estimation. Given a sample $X_1, \dots, X_N$ drawn independently from a random variable X on $\mathbb{R}^d$ with a density f, the goal is to build an estimator $\hat{f}$ of f. This question finds many applications and density estimation is used as a building block for other learning tasks such as clustering ([Chazal et al., 2013], [Campello et al., 2013]) or anomaly detection ([Chandola et al., 2009]) among others.

Non Parametric and Parametric Density Estimation The easiest way to do density estimation is to consider parametric models. Here data is supposed to be drawn from a probability distribution known up to a finite dimensional parameter θ. Estimating the density is then equivalent to estimating θ. One

*Université Paris-Saclay, CNRS, Laboratoire de mathématiques d'Orsay, 91405, Orsay, France. Contact : louis.pujol@universite-paris-saclay.fr

example is the centered multivariate Gaussian framework, where the parameter θ is the covariance matrix Σ. An introduction to parametric statistics can be found in [Wasserman, 2004], chapter 9. This approach suffers from a lack of flexibility as it imposes a strong constraint on the model.

At the other end of the spectrum lies nonparametric density estimation. In this framework, densities are no longer considered as members of some finite-dimensional family but are supposed to belong to a set of functions with a given regularity (Lipschitz or Hölder for example). An introduction to the subject can be found in [Tsybakov, 2004].

Curse of Dimensionality When dealing with multidimensional data, one must be aware of the issues that the number of features can imply. The complexity of a statistical problem can be evaluated through minimax risk study, quantifying the statistical error in a worst-case scenario. In the covariance estimation problem, without further assumption on the covariance matrix, the minimax risk under the Frobenius norm is proportional to $\frac{d}{N}$ (see [Cai et al., 2010]). In the nonparametric framework, the minimax rate is influenced by two parameters: a regularity parameter β and dimension d , rate of convergence for the squared L_2 loss is typically proportional to $N^{\frac{-2\beta}{2\beta+d}}$ (see [Goldenshluger and Lepski, 2012] for an exhaustive coverage of the topic).

We remark that in both situations the dependence on the dimension is adversarial, the higher d, the harder the density estimation problem. This phenomenon is a manifestation of the so-called curse of dimensionality. For practitioners it means that it should be adventurous to use a multivariate density estimator if the sample size is limited and the dimension becomes large, especially in the case of nonparametric estimation. A solution is to assume that unknown densities belong to a class of structured ones.

Structural Density Estimation with Undirected Graphical Models A way to consider structure for a multivariate random variable is to study its undirected graphical model (introduction to the field can be found in [Giraud, 2021] and more in-depth cover in [Wainwright and Jordan, 2008]). As we will not consider directed graphical models, we do not precise that graphs are undirected in the sequel. Given a graph $G = (V, E)$ whose vertices correspond to the features $(X^1, \dots, X^d)$ we say that G is a graphical model for X if the following condition is true:

$$(i, j) \notin E \Leftrightarrow X^i \perp\!\!\!\perp X^j | (X^k)_{k \notin (i,j)}. \tag{1}$$

Constraints on the graph associated with a distribution impose a structure on the density and such a structure can help overcome the curse of dimensionality. However learning a graphical model is known to be a complex task in a lot of situations. An exception is the Gaussian framework, where data distribution is a multivariate normal $\mathcal{N}(0, \Sigma)$. Here the estimation of the graphical model is possible through estimation of the inverse of Σ. This domain is known as Gaussian graphical model (GGM). Different methods are available: graphical lasso [Friedman et al., 2008] which imposes a sparse structure for the graph is probably the most famous example.

In a fully nonparametric setting, where we want to assume as little as we can on the density, one method is available: Forest Density Estimation (FDE)

[Liu et al., 2011] which is limited to graphs without cycles.

Independence Structure In the present work, we focus on the model of independence structure for multivariate density developed by [Lepski, 2013] and studied by [Rebelles, 2015]. It contains d-dimensional densities which can be decomposed as a product of some of these marginals, forming a partition of the original features. In terms of graphical model, it corresponds to graphs which are composed of disjoint fully-connected cliques.

They showed that if the density is assumed to enjoy the property that the size of the biggest block of the partition is equal to $k < d$, then the complexity of density estimation, measured through minimax rate is no longer related to the ambient dimension d but rather to k.

However, these works rely on the analysis of estimators which are not practically computable.

Moderately High Dimension Setting In recent years, a lot of attention was put on high dimensional problems, where the number of features can vary from hundreds to thousands. These applications motivated the introduction of techniques that find a way to drastically reduce the dimension of the data through appropriate transformations. We are interested here in situations of moderately high dimension, where the number of features is lesser or equal than 15. This is of particular interest to distinguish both paradigms as we will develop algorithmic solutions that allow exhaustive search over admissible structures in moderately high dimension but become too time consuming in high dimension.

Our Contribution We developed Independence Structure Density Estimation (ISDE), a method designed to simultaneously compute a partition of the features and a density estimation relying on this partition. Our method enjoys reasonable running time for moderately high dimensional problems and is designed to be combined with any density estimation technique, so it covers parametric as well as nonparametric settings.

For GGM an analogue algorithm already exists [Devijver and Gallopin, 2018], but up to our knowledge, we are the first to design an algorithm to deal with nonparametric density estimation under the model of independence structure.

Organization of the Paper In Section 2 we briefly review existing work about independence structure. In Section 3 we present our algorithm to perform density estimation and independence structure selection. In Section 4 we compare our method with some existing ones for the task of density estimation under Kullback-Leibler loss before analysing its running time in Section 5.

Notations Set_d denotes the set of subsets of $\{1, \ldots, d\}$ and S_d its cardinal. Part_d is the set of all partitions of $\{1, \ldots, d\}$ and B_d its cardinal. If $x = (x_1, \ldots, x_d)$ is a d-dimensional vector and $S \in \mathrm{Set}_d$, $x_S := (x_i)_{i \in S}$. If f is a d-dimensional function, $f_S(x) := \int f(x) \prod_{i \notin S} dx_i$. In order to highlight the fact that $f_S(x)$ does not depend on $(x_i)_{i \notin S}$, we write $f_S(x_S)$ instead of $f_S(x)$.

2 INDEPENDENCE STRUCTURE

In this section we review some theory about nonparametric density estimation and independence structure model.

Minimax Risks for d-dimensional Density Estimation Let $X_1, \dots, X_N$ be *iid* realizations of a random variable in $\mathbb{R}^d$ admitting a density f. The goal of density estimation is to construct an estimator $\hat{f}$ of the density. The hardness of such an estimation task can be assessed using the minimax framework. Assume that the true density belongs to some known model $\mathcal{F}$ and let D be a (pseudo)distance on $\mathcal{F}$, the minimax risk is defined as follows:

$$\inf_{\hat{f}} \sup_{f \in \mathcal{F}} \mathbb{E}\left[D(f, \hat{f})\right] =: \mathcal{R}(D, \mathcal{F}) \tag{2}$$

where the infimum is taken over all measurable functions from the data to $\mathcal{F}$. More specifically, a great part of the literature on the topic deals with the asymptotic regime of $\mathcal{R}(D, \mathcal{F})$ with respect to N.

For $\beta, L, p > 0$ Let us consider $\mathcal{F} = \mathcal{N}p, d(\beta, L)$ the isotropic Nikolsk'ii class over $\mathbb{R}^d$. A probability density g belongs to $\mathcal{F} = \mathcal{N}p, d(\beta, L)$ if the following conditions are fulfilled:

1. $\left\|D_k^i g\right\|_p \leq L \quad \forall k \in 0, \dots, \lfloor\beta\rfloor,\ \forall i \in \{1, \dots, d\}$
2. $\left\|D_i^{\lfloor\beta\rfloor} g(. + e_i) - D_i^{\lfloor\beta\rfloor} g(.)\right\|_p \leq L|t|^{\beta - \lfloor\beta\rfloor} \quad \forall t \in \mathbb{R},\ \forall i \in \{1, \dots, d\}$

where $D_k^i g$ is the kth partial derivative of g with respect to the ith variable and $\lfloor\beta\rfloor$ is the only integer satisfying $\beta - 1 \leq \lfloor\beta\rfloor < \beta$.

The minimax rate of this family of functions was studied in [Hasminskii et al., 1990] considering L_p distances. In particular, the result with the squared L_2 distance is the following:

$$\mathcal{R}\left(\|.\|_2^2, \mathcal{N}p, d(\beta, L)\right) \sim N^{-\frac{2\beta}{2\beta + d}}. \tag{3}$$

As explained in introduction, this bound can be interpreted as a manifestation of the curse of dimensionality. To overcome it, a solution is to consider the independence structure model introduced in [Lepski, 2013].

Independence Structure For $k \leq d$, we define a family of functions:

$$\mathcal{D}_d^k = \left\{ f : \mathbb{R}^d \to \mathbb{R} |\ \exists \mathcal{P} \in \mathrm{Part}_d^k : f(x) = \prod_{S \in \mathcal{P}} f_S(x_S) \right\}.$$

Where Part_d^k is the set of partitions of $\{1, \dots, d\}$ with blocks of size lesser than k. In probabilistic term, a density f over $\mathbb{R}^d$ belongs to $\mathcal{D}_d^k$ if these features can be grouped into independent blocks. Rebelles [Rebelles, 2015] showed that:

$$\mathrm{R}\left(\|.\|_2, \mathcal{N}p, d(\beta, L) \cap \mathcal{D}_d^k\right) \sim N^{-\frac{2\beta}{2\beta + k}}. \tag{4}$$

The striking fact here is that the hardness of the estimation problem is no longer related to the ambient dimension but rather on the size of the biggest block of the partition on which the density function can be decomposed.

Practical Consideration In order to compute minimax rates, [Rebelles, 2015] designed an appropriate estimator, unfortunately, it is not practically computable. However we believe that independence structure model could be of practical interest as it leads to qualitative information about data through the independence structure and tackles the curse of dimensionality.

3 ISDE

In this section we present ISDE, our algorithm designed to perform simultaneously density estimation and independence partition selection in moderately high dimensional setting. Our aim is to provide a method taking point cloud as input and outputting an independence structure (a partition of the features) and a density estimator as a product of marginal estimators.

Hyperparameters Optimization Let Θ denotes a hyperparameters space adapted to our problem. A parameter $\theta \in \Theta$ corresponds to a collection of partition-indexed parameters $(\theta_{\mathcal{P}})_{\mathcal{P}\in \mathrm{Part}_d}$, each of them being a list of parameter for marginal density estimates: $\theta_{\mathcal{P}} = (\theta_{\mathcal{P}}(S))_{S\in\mathcal{P}}$. Then to each $\theta \in \Theta$ is associated a family of density estimators satisfying independence structure condition:

$$\left(\hat{f}^{\theta}_{\mathcal{P}}\right)_{\mathcal{P}\in \mathrm{Part}_d} = \left(\prod_{S\in\mathcal{P}} \hat{f}^{\theta_{\mathcal{P}}(S)}_{S}\right)_{\mathcal{P}\in \mathrm{Part}_d}. \tag{5}$$

We do not precise here which set of hyperparameters Θ we take. This choice will be an input of ISDE as we want our method to be usable indifferently with any local density estimator.

Number of Partitions and Complexity Bottleneck Apparently, we need to compute a density estimation of the form $\prod_{S\in\mathcal{P}} \hat{f}^{\theta_{\mathcal{P}}(S)}_{S}$ for all $\mathcal{P} \in \mathrm{Part}_d$. But as we will see, the number of partitions is rapidly prohibitive even for moderate dimensions. Then we need to avoid this complexity bottleneck by designing an algorithm where the optimization tasks have to be made over Set_d.

Let us start by comparing S_d and B_d, the respective cardinals of Set_d and Part_d. We have $S_d = 2^d - 1$ and B_d is known as the Bell number of order d. Table 1 shows how these quantities compare for moderately high dimensions.

d	10	11	12	13	14	15
S_d	$1,023$	$2,047$	$4,095$	$8,191$	$16,383$	$32,767$
B_d	$21,147$	$115,975$	$678,570$	$4,213,597$	$27,644,437$	$190,899,322$

Table 1: Number of Partitions vs Number of Subsets

This indicates that it would be beneficial to find a way to avoid the computation of B_d estimators. Intuitively, as estimators are built as combination of marginals estimators, it seems reasonable to decouple marginal estimations from partition selection. We will now see that this idea can be implemented through an appropriate choice of loss function.

Choice of Loss Function Though theory about independence structure was developed for L_p losses, we found it more convenient to rephrase the estimation problem using Kullback-Leibler (KL) divergence as discrepancy measure. The reason is because as KL involves log-densities, it is well suited for densities in $\mathcal{D}_d^k$ as the logarithm of a product of marginal densities becomes the sum of the marginal log-densities.

For an estimator $\hat{f}$, the Kullback-Leibler divergence is defined as follows:

$$\mathrm{KL}\left(f\|\hat{f}\right) = P\left[\log\left(\frac{f}{\hat{f}}\right)\right] = P\left[\log\left(f\right)\right] - P\left[\log(\hat{f})\right] \tag{6}$$

where for any function g, $P[g] = \int g(x)f(x)dx$. We see that minimizing $\mathrm{KL}\left(f\|\hat{f}\right)$ is equivalent to maximizing $P\left[\log(\hat{f})\right]$.

Optimization Problem Then, the optimization problem we want to solve rewrites as follows:

$$\max_{\mathcal{P}\in\mathrm{Part}_d,\theta\in\Theta} P\left[\log(\hat{f}_{\mathcal{P}}^{\theta})\right] \tag{7}$$

$$= \max_{\mathcal{P}\in\mathrm{Part}_d,\theta\in\Theta} \sum_{S\in\mathcal{P}} P\left[\log(\hat{f}_S^{\theta_{\mathcal{P}}(S)})\right]. \tag{8}$$

With this formulation, it appears that the optimization of $\theta_{\mathcal{P}}$ can be done through independent optimizations of the parameters $(\theta_{\mathcal{P}}(S))_{S\in\mathcal{P}}$. What's more, if the same subset S is shared by two partitions $\mathcal{P}$ and $\mathcal{P}'$ we have:

$$\arg\max_{\theta_{\mathcal{P}}(S)} P\left[\hat{f}_S^{\theta_{\mathcal{P}}(S)}\right] = \arg\max_{\theta_{\mathcal{P}'}(S)} P\left[\hat{f}_S^{\theta_{\mathcal{P}'}(S)}\right]. \tag{9}$$

Then it is only necessary to consider a hyperparameter space indexed by Set_d: $\Theta = (\theta(S))_{S\in\mathrm{Set}_d}$. We can rewrite the optimization task as follows:

$$\max_{\mathcal{P}\in\mathrm{Part}_d} \sum_{S\in\mathcal{P}} \left\{\max_{\theta\in\Theta} P\left[\log(\hat{f}_S^{\theta(S)})\right]\right\}. \tag{10}$$

Then under KL loss, hyperparameters optimization can be made over Set_d instead of Part_d, as highlighted in Table 1, this leads to an appreciable gain in terms of algorithmic complexity.

Empirical Formulation of the Optimization Problem The rephrasing above indicates that under KL loss, hyperparameters optimization and partition selection become two separated tasks. This incites us to design an algorithm consisting of two steps: first compute a marginal estimation for all subsets of features and then find the best combination of these. Unfortunately, the optimization problem Eq. (10) cannot be solved directly as it requires the knowledge of P. Here we explain how we construct an empirical version of it.

Let n and m be two positive integers such that $m + n = N$. The dataset $X_1, \ldots, X_N$ is split into two disjoint subsamples:

- $W_1, \ldots, W_m$ used to compute marginal estimators $(\hat{f}_S)_{S \in \mathrm{Set}_d}$
- $Z_1, \ldots, Z_n$ used to compute empirical log-likelihoods $(\ell_n(S))_{S \in \mathrm{Set}_d}$ for these estimators. We have $\ell_n(S) = \frac{1}{n} \sum_{i=1}^{n} \log\left(\hat{f}_S(Z_i)\right)$

Let $\ell_n(\mathcal{P}) = \sum_{S \in \mathcal{P}} \ell_n(S)$, the empirical optimization task can be written as:

$$\max_{\mathcal{P} \in \mathrm{Part}_d} \ell_n(\mathcal{P}) = \max_{\mathcal{P} \in \mathrm{Part}_d} \sum_{S \in \mathcal{P}} \ell_n(S). \tag{11}$$

Partition Selection A naive approach to solve it is to compute $\ell_n(\mathcal{P})$ for every partition of Part_d and then find the optimal one. However we will see that this approach becomes time consuming as d grows. It will therefore be appreciable to find a reformulation of this optimization speeding up computation. It is possible through an linear programming (LP) under constraints reformulation of Eq. (11):

Solve:

$$\max_{x \in \mathbb{R}^{S_d}} \sum_{S \in \mathrm{Set}_d} \ell_n(S) x(S) \tag{12}$$

Under constraints:

$$Ax = (1, \ldots, 1)^{\mathrm{T}} \tag{13}$$

$$x \in \{0, 1\}^{S_d} \tag{14}$$

Where x is a binary vector representing which elements of Set_d are selected and A is a $d \times S_d$ matrix where each column is a binary vector representing composition of one of the sets of Set_d:

$$A = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\ 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 1 \\ 0 & 0 & 1 & \ldots & 0 & 0 & 1 & \ldots & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 \end{pmatrix}$$

the condition $Ax = (1, \ldots, 1)^{\mathrm{T}}$ then ensures that each feature is chosen once, implying that the sets selected through x form a partition.

We validate this approach through running time comparison (Section 3) between implementation of a naive approach and a linear program solver. In this experiment the quantities $(\ell_n(S))_{S \in \mathrm{Set}_d}$ are fixed, the naive approach consists in a for loop (implemented here in Python), computing $\ell_n(\mathcal{P})$ for all $\mathcal{P} \in \mathrm{Part}_d$ and returning the maximum. For the LP formulation, computations are made with the Python package PulP [Mitchell et al., 2011]. With the naive formulation, partition selection takes approximately 3 hours in dimension 15 but less than 10 seconds with LP formulation.

d	9	10	11	12	13	14	15
Naive Formulation	0.2	0.9	5.2	32.5	219.9	1304.4	10437.5
LP Formulation	0.1	0.2	0.4	0.8	1.9	4.1	9.1

Table 2: Running Times (seconds): Linear Programming vs Naive Approach for Partition Selection

Conclusion The resulting algorithm is algorithm 1. It enjoys the following properties:

- It is exhaustive: it tests all B_d possible partitions of variables even if it only requires the computation of $\text{Set}_d = 2^d - 1$ marginal estimators
- It is versatile: it can be used with any multivariate density estimation algorithm as an input

input : $X_1, \dots, X_N \in [0,1]^d$, integers m and n and a subroutine to perform multidimensional density estimation
output: Independence structure $\hat{\mathcal{P}}$, marginal estimates $(\hat{f}_S)_{S \in \hat{\mathcal{P}}}$
begin
  for $S \in \text{Set}_d$ **do**
    Compute $\hat{f}_S(W_1, \dots, W_m)$ thanks to the density estimation subroutine
    Compute $\ell_n(S)$
  end
  Compute $\hat{\mathcal{P}} \in \underset{\mathcal{P} \in \text{Part}_d}{\arg\max} \sum_{S \in \mathcal{P}} \ell_n(S)$ using linear programming formulation
end

Algorithm 1: ISDE

4 SIMULATIONS

In this section we show the performance of ISDE on simulated data satisfying independence structure. To illustrate the versatility of our method we apply it in two scenarios: Gaussian framework and nonparametric framework.

4.1 Gaussian Data with Independence Structure

Data Generating Process The Gaussian Graphical Models (GGM) theory indicates that edges of the undirected graphical model associated with a Gaussian distribution $\mathcal{N}(0, \Sigma)$ are the non-zero entries of the precision matrix Σ^{-1}. As inverse preserves block-diagonal structure, we can easily simulate data from Gaussian with independence structure.

For a positive integer s and a real number $\sigma \in (0,1)$ we denotes by Σ_σ^s the $s \times s$ matrix whose diagonal entries are 1 and nondiagonal entries are σ. Then for a list of positive integers $S = [s_1, \dots s_K]$ we define the block diagonal matrix:

$$\Sigma_\sigma^S = \left(\begin{array}{c|c|c|c} \Sigma_\sigma^{s_1} & \mathbf{0} & \dots & \mathbf{0} \\ \hline \mathbf{0} & \Sigma_\sigma^{s_2} & \ddots & \vdots \\ \hline \vdots & \ddots & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \dots & \mathbf{0} & \Sigma_\sigma^{s_K} \end{array}\right) \tag{15}$$

The distribution $\mathcal{N}\left(0, \Sigma_\sigma^S\right)$ satisfies the independence structure condition with partition $\left(\left\{\sum_{i=1}^{j-1} s_i + 1, \dots, \sum_{i=1}^{j} s_i\right\}\right)_{j=1,\dots,K}$.

Evaluation Scheme If $\hat{\Sigma}$ and Σ are respectively the estimated and the true covariance, the Kullback-Leibler risk can be explicitly computed:

$$\mathrm{KL}\left(\mathcal{N}(0,\Sigma) \| \mathcal{N}(0,\hat{\Sigma})\right) = \sum_{v \in \mathrm{Sp}(A)} \frac{v - \log(1+v)}{2} \tag{16}$$

where $A = (\hat{\Sigma}^{-1} - \Sigma^{-1})\Sigma$.

Benchmarked Methods Two methods will be compared to ISDE for the task of covariance estimation.

The first estimator is the simple **Empirical Covariance**, which is the maximum likelihood estimator in this setting. The second estimator is **Block-Diagonal Covariance Selection** (BDCS) developed in [Devijver and Gallopin, 2018]. It is designed to estimate an independence structure in the context of GGM. This algorithm works in two steps:

- Compute a family of nested partitions candidates to be the independence structure
- Choose a partition in this family using a slope heuristic approach

More details can be found in the original paper. Up to our knowledge this is the only work to deal specifically with independence structure in the GGM framework.

ISDE Inputs We run algorithm 1 with $m = n = 0.5 \times N$ and simple empirical covariance as multivariate density estimator.

Performance We compare the three methods described above for fixed σ, N and different structures S. More results can be found in appendix. Results in terms of KL loss are collected in Table 3. Each experiment is repeated 10 times and the scores displayed are the mean KL losses and standard deviation over these repetitions.

S	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3 ,2]
ISDE	0.41 ± 0.24	1.85 ± 0.76	3.08 ± 1.20	$\mathbf{3.76 \pm 1.42}$
BDCS	$\mathbf{0.34 \pm 0.22}$	$\mathbf{1.81 \pm 0.69}$	$\mathbf{2.57 \pm 1.14}$	4.06 ± 2.57
Empirical	0.65 ± 0.28	3.75 ± 0.75	6.83 ± 1.23	11.61 ± 1.75

Table 3: Gaussian: KL Losses ($.10^3$) - $\sigma = 0.7$, $N = 6000$

Recovery We are interested not only in performance but we also want to find the right partition in order to give qualitative insight about datasets. In Table 4 we collect, for the same experiment as above, the rate of recovery of the true partition. In parentheses is displayed the rate of admissible output partition: a partition is said to be admissible if all the blocks of the original partition are subsets of blocks of this one.

S	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3 ,2]
ISDE	90%(100%)	80%(100%)	30%(100%)	40%(100%)
DG	100%(100%)	90%(100%)	70%(100%)	70%(100%)

Table 4: Gaussian: Recovery - $\sigma = 0.7$, $N = 6000$

Conclusion We remark that ISDE and BDCS give almost similar results for these data both in terms of performance and recovery and that the bigger the dimension, the better they are comparatively to a naive empirical covariance approach.

We want to highlight a difference between ISDE and BDCS. BDCS starts by selecting a family of up to d nested partitions and then selects among them. This approach uses a preliminary covariance estimator in order to design this family of nested partitions, this is reasonable as for Gaussian data multidimensional dependencies between features are entirely determined by pairwise dependencies. Outside the scope of GGM this approach does not remain valid as features of a random variable can be pairwise independent but mutually dependent. ISDE can handle more general settings as it selects among all partitions.

4.2 Nonparametric Data with Independence Structure

Data Generating Process For a given structure $S = [s_1, \ldots, s_K]$, the data generating process is defined as follows. For each $s_i \in S$, we define a s_i dimensional dataset drawn from P_i:

- If $s_i = 1$, P_i is the uniform distribution over $[0, 1]$
- If $s_i = 2$, P_i is a distribution corresponding to data sample near two concentric circles with different radii
- If $s_i = 3$, a sample X from P_i is obtained as follows: let Y_1 and Y_2 be two independent Bernoulli variables with probability of success 0.5 and $Y_3 = |Y_1 - Y_2|$. X is then drawn from the multivariate Gaussian distribution $\mathcal{N}((Y_1, Y_2, Y_3), 0.08 \times I_3)$. This is a situation where features of P_i are pairwise independent but not mutually independent

- If $s_i \geq 4$, P_i is a mixture of two multivariate Gaussian distributions, one centered in $(0, \ldots, 0)$, the other in $(1, \ldots, 1)$

The dataset is then defined as the concatenation of these, plus a rescaling of each feature in order that it values lies between 0 and 1 (it is necessary in order to use FDE, define later on). This rescaling step does not affect the independence structure as it is done featurewise.

Evaluation Scheme Here the Kullback-Leibler loss between true density f and an estimator $\hat{f}$ is not computable. In order to evaluate the performance of an estimator, we compute the negative log-likelihood on a validation set $X^{\text{valid}} = X_1^{\text{valid}}, \ldots, X_M^{\text{valid}}$ drawn independently from the same distribution as $X_1, \ldots, X_N$:

$$\text{Score}(\hat{f}) = \frac{1}{M} \sum_{i=1}^{M} -\log\left(\hat{f}\left(X_i^{\text{valid}}\right)\right). \tag{17}$$

The set $X^{\text{valid}} = X_1^{\text{valid}}, \ldots, X_M^{\text{valid}}$ is never used to estimate $\hat{f}$. In the experiments below, we set $M = 5000$.

Benchmarked Methods Two estimators will be compared to ISDE for the task of nonparametric multivariate density estimation.

The first one is **Cross-Validated Kernel Density Estimator** (CVKDE) with Gaussian kernel. For a given bandwidth $h > 0$ we define:

$$\hat{f}_h(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{\exp\left(-\frac{(X_i - x)^{\mathrm{T}}(X_i - x)}{2h^d}\right)}{(2\pi)^{d/2} h^d}. \tag{18}$$

The final estimator is $\hat{f}_{\hat{h}}$ where $\hat{h}$ is selected through a cross-validation scheme in order to minimize negative log-likelihood. We ran our experiments with a 5-fold cross validation and tested bandwidths belong to a regular grid on a log-scale from 0.01 to 1 with 50 elements.

The second one is **Forest Density Estimation** (FDE) developed by Liu *et al.* [Liu et al., 2011], designed to estimate a graphical model for nonparametric densities. The estimated graph is a forest, a graph without cycle. In this case the density can be expressed only through 1 and 2-dimensional marginals. If $G = (V, E)$ is a forest, a random variable with density f admitting G as graphical model can be written as follows:

$$f(x) = \prod_{(i,j) \in E} \frac{f(x_i, x_j)}{f(x_i) f(x_j)} \prod_{k=1}^{d} f(x_k). \tag{19}$$

Estimating such a density only requires the estimation of marginals up to dimension 2, they consider estimators of the form Eq. (19). Then for each couple of features, a score is computed quantifying the loss of information induced by assuming they are independent. After that, a preliminary tree (connected graph without cycle) is constructed using Kruskal's algorithm and then the tree is pruned using held-out data. More details can be found in the original paper. As authors do not provide an empirical bandwidth selection method, we had to plug a value, we chose $h = 0.05$ after some tries on our data.

ISDE Inputs We run algorithm 1 with $m = n = 0.5 \times N$. For density estimation subroutine, we tested two options: CVKDE as presented above (ISDE_CVDE) and kernel density estimator with Gaussian kernel with a fixed h equals to 0.05 (ISDE_Fixed_h).

Performance Table 5 shows empirical negative log-likelihood for methods listed above and for different structures.

	[2, 2, 1]	[3, 3, 3]	[4, 4, 2, 2]
ISDE_CVKDE	$\mathbf{-1.79 \pm 0.09}$	$\mathbf{-3.918 \pm 0.139}$	$\mathbf{-6.44 \pm 0.16}$
ISDE_Fixed_h	-1.00 ± 0.02	$\mathbf{-3.915 \pm 0.136}$	-5.69 ± 0.16
FDE	-1.00 ± 0.02	-2.74 ± 0.09	-5.30 ± 0.14
CVKDE	-0.56 ± 0.02	-3.39 ± 0.11	-4.06 ± 0.12

Table 5: Nonparametric: Empirical Negative Log-Likelihood - $N = 5000$

Recovery Table 6 shows the recovery rates of the independence structure for ISDE_CVDE and ISDE_Fixed_h.

	[2, 2, 1]	[3, 3, 3]	[4, 4, 2, 2]
ISDE_CVDE	100%(100%)	100%(100%)	100%(100%)
ISDE_Fixed_h	100%(100%)	100%(100%)	100%(100%)

Table 6: Nonparametric: Recovery - $N = 5000$

Conclusion For the structure $[2, 2, 1]$, ISDE_Fixed_h and FDE give similar results as they output the same graph and use the same bandwidth. ISDE_CVKDE has better results thanks to bandwidths optimization.

For $[3, 3, 3]$, as features are pairwise independent, FDE outputs at every try a graph without any edge and computes the density as a product of one-dimensional marginals, leading to poor results in comparison to ISDE_CVKDE and ISDE_Fixed_h. Here no difference is observed between ISDE_Fixed_h and ISDE_CVKDE, this is because here, selected bandwidths are close to the fixed one.

For $[4, 4, 2, 2]$, at every try, FDE outputs a subgraph of the true graphical model. It leads to better estimation than CVKDE but worse than ISDE_CVDE and ISDE_Fixed_h which learns the true independence structure at every try.

Thus, ISDE_Fixed_h and ISDE_CVKDE lead to better results than FDE for the task of structured density estimation under KL loss when independence structure is met. ISDE_CVKDE outperforms ISDE_Fixed_h as it optimizes over a set of bandwidths for every marginal estimator. The recovery study indicates that independence structure can be learned using indifferently ISDE_Fixed_h or ISDE_CVKDE.

5 RUNNING TIME

In this section we analyze the running time of ISDE for moderately high dimensional data. All experiments were run on a laptop with the following hardware, CPU: Intel Core i7-9850H CPU @ 2.60GHz and GPU: Nvidia Quadro P620. All computations involving Gaussian kernels have been performed on GPU using the KeOps package.

The running time of algorithm 1 is influenced by the subroutine for marginal density estimations and by the parameters d, m and n. Here we give running times for experimental settings described in Section 4.

For GGM scenario, we considered empirical covariance as subroutine for multivariate density estimation. Table 7 shows running time of ISDE for the parameters described in Section 4.

d	4	9	12	16
Time (seconds)	0.026	0.333	2.923	70.06

Table 7: Mean Running Times: ISDE with Empirical Covariance

For the nonparametric scenario, we used both CVKDE and a Gaussian KDE with fixed bandwidth as subroutines for multivariate density estimation. Table 8 shows the running time of ISDE_CVKDE and ISDE_Fixed_h for the parameters described in Section 4:

d	5	9	12	15
ISDE_CVDE	80.05	1583.9	14487.0	
ISDE_Fixed_h	1.1	21.9	200.9	1837.6

Table 8: Mean Running Times (seconds): ISDE_CVKDE and ISDE_Fixed_h

For $m = n = 2500$ and in dimension 15, ISDE_Fixed_h runs in approximately 30 minutes. ISDE_CVKDE (5-fold cross validation over 50 parameters) runs in more than 4 hours in dimension 12. To cover the moderately high dimensional setting keeping running times reasonable, it is therefore necessary to use ISDE_Fixed_h instead of ISDE_CVKDE if data has more than 11 or 12 features. Running times for more choices of parameters can be found in appendix.

6 CONCLUSION

Take-Home Message We designed ISDE, an algorithm outputting an independence structure and an estimation of the density taking this structure into account for data in moderately high dimension (up to 15 features). To design it we had to overcome two difficulties. We reduced the quantity of hyperparameters to learn with an appropriate choice of loss function and made the partition selection step feasible in acceptable time through linear programming reformulation.

ISDE is versatile, it takes any basic multidimensional density estimator as input so it can be used in parametric as well as in nonparametric frameworks and exhaustive as it searches over all partitions of features. Up to our knowledge we

are the first to propose a method taking into account independence structure in the context of nonparametric density estimation with kernel density estimators.

We proved its performance on simulated data, in GGM with independence structure and in nonparametric independence structure framework. This performance was measured both in term of KL loss in comparison to other methods and in term of independence structure recovery. We proved that it runs in reasonable time for moderately high dimensions, if inputs are well chosen with respect to the size of the dataset.

We showed in particular that it runs in approximately 30 minutes (on a laptop equipped with a GPU) in dimension 15 with Gaussian kernel density estimator at the price of using fixed bandwidth. But results of Section 4 indicates that we can even recover the true independence structure with this choice. We suggest to do so as a first step on real data and then to optimize bandwidths for outputted subsets of $\{1, \dots, d\}$.

We do not pretend that ISDE is meaningful on every density estimation situations. We proved that if the number of features does not exceed 15 and if the independence structure is met then it is relevant to use it. But in other dependence structures, other methods may be more relevant. For applications, we recommend to empirically compare ISDE with other algorithms designed to identify other structures and to select among them.

Code Availability The code to reproduce the experiments presented here is available at https://github.com/Louis-Pujol/ISDE-Paper.

Acknowledgement The author is thankful to Marc Glisse[1] and Pascal Massart[2] for their constructive remarks on this work.

Funding This research work is funded through the program Paris Region PhD of DIM Mathinnov.

References

[Cai et al., 2010] Cai, T. T., Zhang, C.-H., and Zhou, H. H. (2010). Optimal rates of convergence for covariance matrix estimation. *The Annals of Statistics*, 38(4):2118–2144.

[Campello et al., 2013] Campello, R. J., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

[Chazal et al., 2013] Chazal, F., Guibas, L. J., Oudot, S. Y., and Skraba, P. (2013). Persistence-based clustering in riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):1–38.

[1] Inria Saclay
[2] Université Paris-Saclay

[Devijver and Gallopin, 2018] Devijver, E. and Gallopin, M. (2018). Block-diagonal covariance selection for high-dimensional gaussian graphical models. *Journal of the American Statistical Association*, 113(521):306–314.

[Friedman et al., 2008] Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

[Giraud, 2021] Giraud, C. (2021). *Introduction to high-dimensional statistics*. Chapman and Hall/CRC.

[Goldenshluger and Lepski, 2012] Goldenshluger, A. and Lepski, O. (2012). On adaptive minimax density estimation on rd. *arXiv preprint arXiv:1210.1715*.

[Hasminskii et al., 1990] Hasminskii, R., Ibragimov, I., et al. (1990). On density estimation in the view of kolmogorov's ideas in approximation theory. *The Annals of Statistics*, 18(3):999–1010.

[Lepski, 2013] Lepski, O. (2013). Multivariate density estimation under sup-norm loss: oracle approach, adaptation and independence structure. *Annals of Statistics*, 41(2):1005–1034.

[Liu et al., 2011] Liu, H., Xu, M., Gu, H., Gupta, A., Lafferty, J., and Wasserman, L. (2011). Forest density estimation. *The Journal of Machine Learning Research*, 12:907–951.

[Mitchell et al., 2011] Mitchell, S., Consulting, S. M., and Dunning, I. (2011). Pulp: A linear programming toolkit for python.

[Rebelles, 2015] Rebelles, G. (2015). Lp adaptive estimation of an anisotropic density under independence hypothesis. *Electronic journal of statistics*, 9(1):106–134.

[Tsybakov, 2004] Tsybakov, A. B. (2004). Introduction to nonparametric estimation, 2009. *URL https://doi. org/10.1007/b13794. Revised and extended from the*, 9:10.

[Wainwright and Jordan, 2008] Wainwright, M. J. and Jordan, M. I. (2008). *Graphical models, exponential families, and variational inference*. Now Publishers Inc.

[Wasserman, 2004] Wasserman, L. (2004). *All of statistics: a concise course in statistical inference*, volume 26. Springer.

Appendix

In order to help people using our code, we present an exhaustive running time study for different inputs of ISDE. The running times presented here do not take into account the execution time of partition selection step, this can be found in the Table 2 of the paper. Code is available to reproduce these experiments and get results with different hardware. We used the following hardware CPU: Intel Core i7-9850H CPU @ 2.60GHz and GPU: Nvidia Quadro P620. All computations involving Gaussian kernels have been performed on GPU using the KeOps package.

ISDE With Fixed Bandwidth KDE

Table 9 to Table 22 correspond to ISDE with a fixed bandwidth Gaussian KDE.

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
2500	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2
5000	0.0	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
10000	0.1	0.2	0.2	0.3	0.3	0.4	0.5	0.5	0.6	0.6

Table 9: Estimated Running Times (seconds): ISDE_Fixed_h, d=2

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.2
2500	0.0	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.4	0.4
5000	0.1	0.2	0.2	0.3	0.4	0.5	0.6	0.6	0.7	0.8
10000	0.2	0.3	0.5	0.6	0.8	0.9	1.0	1.3	1.4	1.6

Table 10: Estimated Running Times (seconds): ISDE_Fixed_h, d=3

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
2500	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
5000	0.2	0.4	0.5	0.7	0.9	1.1	1.3	1.4	1.6	1.8
10000	0.4	0.7	1.1	1.4	1.7	2.1	2.4	2.8	3.2	3.4

Table 11: Estimated Running Times (seconds): ISDE_Fixed_h, d=4

ISDE with CVKDE

Table 23 to Table 36 correspond to ISDE with a fixed bandwidth Gaussian KDE.

ISDE with Empirical Covariance

Table 37 to Table 47 correspond to ISDE with Empirical Covariance.

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.1	0.2	0.2	0.3	0.4	0.5	0.5	0.6	0.7	0.8
2500	0.2	0.4	0.7	0.9	1.1	1.3	1.5	1.7	1.9	2.1
5000	0.4	0.8	1.2	1.6	2.0	2.3	2.7	3.1	3.5	3.9
10000	0.8	1.6	2.4	3.2	3.8	4.6	5.2	6.3	7.1	7.9

Table 12: Estimated Running Times (seconds): ISDE_Fixed_h, d=5

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.2	0.3	0.5	0.7	0.8	1.0	1.1	1.3	1.4	1.6
2500	0.5	1.0	1.4	1.9	2.4	2.8	3.3	3.7	4.2	4.7
5000	0.9	1.7	2.6	3.4	4.3	5.1	5.9	6.9	7.6	8.6
10000	1.9	3.3	5.1	6.9	8.5	10.0	11.6	13.7	14.9	16.9

Table 13: Estimated Running Times (seconds): ISDE_Fixed_h, d=6

7 MORE EXPERIMENTS

Here we reproduce the experiments of Gaussian simulations section with different values for N and σ. Conclusions on the relative quality of tested estimators remain the same as in the paper.

Different Values of σ

Table 48 to Table 52 correspond to different values of σ for $N = 6000$

Different Values of N

Table 53 to Table 57 correspond to different values of N for $\sigma = 0.7$.

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.4	0.8	1.1	1.4	1.7	2.1	2.4	2.7	3.0	3.4
2500	1.1	2.1	3.0	4.0	5.0	6.0	7.0	7.9	8.9	9.8
5000	1.9	3.7	5.5	7.4	9.1	10.9	12.7	14.7	16.2	18.0
10000	3.7	7.1	10.8	14.2	17.3	22.1	25.9	28.3	31.8	34.8

Table 14: Estimated Running Times (seconds): ISDE_Fixed_h, d=7

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.9	1.6	2.3	3.0	3.7	4.4	5.1	5.9	6.5	7.2
2500	2.2	4.3	6.5	8.4	10.5	12.6	14.7	16.8	18.7	20.8
5000	4.1	7.8	11.7	15.4	19.5	23.0	26.8	30.7	34.4	39.1
10000	7.9	15.8	22.4	30.0	38.9	45.2	52.3	60.2	68.6	74.0

Table 15: Estimated Running Times (seconds): ISDE_Fixed_h, d=8

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	1.8	3.4	4.9	6.3	7.9	9.2	10.9	12.3	13.8	15.1
2500	4.7	9.3	13.4	17.8	22.3	26.6	31.0	35.3	39.8	43.9
5000	8.6	16.7	25.4	32.5	40.8	48.9	58.8	64.6	72.7	82.4
10000	16.9	32.0	48.1	62.4	77.8	96.5	111.2	123.8	148.3	160.6

Table 16: Estimated Running Times (seconds): ISDE_Fixed_h, d=9

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	3.8	7.2	9.8	13.2	16.1	19.2	22.2	25.4	28.2	31.2
2500	9.8	19.4	27.6	36.9	45.8	54.8	63.9	72.9	81.9	90.7
5000	18.0	35.0	51.8	67.4	85.6	100.5	117.9	134.7	154.5	170.1
10000	35.4	67.3	96.0	129.1	159.8	193.5	227.1	257.8	289.7	317.7

Table 17: Estimated Running Times (seconds): ISDE_Fixed_h, d=10

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	8.5	14.4	20.5	27.8	33.4	39.7	46.8	52.7	59.0	65.6
2500	21.2	39.4	58.6	76.8	95.7	114.9	133.5	152.7	171.2	190.0
5000	37.0	71.1	108.9	143.5	184.8	214.7	250.4	285.4	321.3	363.3
10000	70.6	142.6	200.4	272.7	341.9	405.8	466.7	524.2	603.8	664.8

Table 18: Estimated Running Times (seconds): ISDE_Fixed_h, d=11

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	16.1	29.7	42.8	56.2	69.6	83.0	96.8	110.6	123.2	137.5
2500	42.1	82.0	121.3	160.4	201.1	239.4	278.9	319.4	357.8	398.5
5000	78.7	149.5	236.8	312.5	379.8	456.9	548.0	605.6	691.2	757.3
10000	145.6	287.0	423.5	562.2	697.4	833.5	982.7	1124.1	1254.1	1403.7

Table 19: Estimated Running Times (seconds): ISDE_Fixed_h, d=12

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	33.0	61.0	88.9	116.6	144.5	175.8	200.4	229.8	256.1	284.1
2500	87.7	171.6	253.4	337.5	419.0	500.7	582.3	663.8	747.1	829.2
5000	167.8	333.9	487.2	642.6	805.3	964.0	1142.9	1283.1	1444.1	1633.5
10000	311.0	592.1	912.7	1172.7	1461.0	1750.7	2063.1	2374.2	2673.7	2918.5

Table 20: Estimated Running Times (seconds): ISDE_Fixed_h, d=13

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	72.4	130.4	184.4	242.8	302.6	358.7	420.0	475.0	533.4	593.3
2500	189.0	354.6	526.0	700.3	870.9	1042.7	1214.3	1385.7	1560.1	1730.9
5000	347.0	678.8	1001.1	1350.7	1698.5	2003.0	2416.3	2761.3	3173.3	3523.4
10000	632.1	1230.0	1841.6	2488.6	3128.0	3679.8	4279.8	4849.6	5531.1	6125.0

Table 21: Estimated Running Times (seconds): ISDE_Fixed_h, d=14

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	141.3	262.8	382.7	503.5	630.3	749.3	866.1	986.7	1107.7	1232.2
2500	380.3	738.9	1097.7	1455.2	1813.2	2174.2	2535.7	2887.9	3249.3	3611.4
5000	732.8	1413.3	2159.3	2842.2	3600.2	4247.8	5086.1	5742.7	6424.6	7145.1
10000	1322.3	2596.4	4039.4	5174.0	6484.9	7830.6	8991.2	10264.7	11426.5	12677.4

Table 22: Estimated Running Times (seconds): ISDE_Fixed_h, d=15

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	1.4	3.0	4.3	5.8	7.3	8.9	10.6	12.3	19.1	21.7
2500	1.4	3.1	4.4	5.8	7.4	8.9	10.7	12.4	19.2	21.8
5000	1.4	3.1	4.4	5.9	7.4	9.0	10.8	12.6	19.3	21.9
10000	1.4	3.1	4.5	6.0	7.6	9.2	11.0	12.8	19.6	22.2

Table 23: Estimated Running Times (seconds): ISDE_CVKDE, d=2

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	3.6	7.4	10.9	14.2	18.2	22.2	26.5	31.7	47.6	53.6
2500	3.7	7.4	11.0	14.3	18.3	22.4	26.7	31.9	47.8	53.9
5000	3.7	7.5	11.1	14.5	18.5	22.6	26.9	32.2	48.2	54.2
10000	3.8	7.7	11.4	14.8	18.8	23.1	27.4	32.8	48.8	54.9

Table 24: Estimated Running Times (seconds): ISDE_CVKDE, d=3

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	7.7	16.6	25.2	33.3	42.5	50.8	58.9	68.4	106.2	121.4
2500	7.8	16.7	25.4	33.5	42.8	51.2	59.4	68.9	106.7	122.0
5000	7.9	16.9	25.7	33.9	43.2	51.7	59.9	69.6	107.5	122.8
10000	8.1	17.2	26.2	34.5	44.1	52.7	61.1	70.9	108.9	124.4

Table 25: Estimated Running Times (seconds): ISDE_CVKDE, d=4

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	15.6	35.4	52.1	69.3	87.7	106.9	130.3	147.9	233.7	259.8
2500	15.8	35.7	52.5	69.8	88.4	107.7	131.2	149.0	234.9	261.2
5000	16.0	36.0	53.0	70.5	89.3	108.8	132.4	150.4	236.5	263.0
10000	16.3	36.8	54.2	71.9	91.1	111.0	135.0	153.1	239.8	266.5

Table 26: Estimated Running Times (seconds): ISDE_CVKDE, d=5

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	32.7	74.9	109.9	147.1	187.7	226.9	269.0	312.5	489.7	551.8
2500	33.0	75.5	110.8	148.3	189.2	228.7	271.1	314.8	492.4	554.7
5000	33.4	76.2	111.9	149.8	191.1	231.0	273.8	317.8	495.7	558.5
10000	34.3	77.8	114.5	153.3	194.9	235.7	279.1	324.3	502.3	566.5

Table 27: Estimated Running Times (seconds): ISDE_CVKDE, d=6

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	67.7	157.3	233.8	310.4	392.7	476.9	568.2	658.8	1029.5	1169.3
2500	68.4	158.6	235.7	313.0	395.9	480.7	572.6	663.9	1035.2	1175.6
5000	69.2	160.2	238.1	316.3	399.9	485.5	578.2	670.4	1042.5	1183.6
10000	70.9	163.5	243.3	323.3	408.6	495.8	590.1	684.2	1056.9	1199.7

Table 28: Estimated Running Times (seconds): ISDE_CVKDE, d=7

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	140.7	330.9	487.9	652.6	823.4	1004.7	1189.9	1381.6	2166.5	2451.3
2500	142.0	333.5	491.9	657.9	830.1	1012.6	1199.3	1392.3	2178.5	2464.7
5000	143.7	336.9	497.0	664.7	838.9	1022.8	1211.8	1406.5	2194.3	2481.6
10000	147.3	344.5	507.5	679.5	856.4	1043.0	1235.0	1433.3	2225.3	2516.2

Table 29: Estimated Running Times (seconds): ISDE_CVKDE, d=8

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	308.2	715.6	1059.4	1398.0	1759.7	2135.5	2585.0	2964.9	4609.4	5249.2
2500	311.0	721.3	1067.9	1409.4	1773.9	2152.4	2604.6	2987.5	4634.8	5277.4
5000	314.9	728.6	1078.8	1424.3	1792.0	2174.1	2629.9	3016.4	4668.4	5314.8
10000	322.6	743.4	1101.8	1453.5	1829.2	2218.6	2680.7	3075.6	4739.2	5386.6

Table 30: Estimated Running Times (seconds): ISDE_CVKDE, d=9

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	610.2	1469.8	2163.9	2945.7	3671.8	4399.3	5179.6	6052.2	9569.6	10738.2
2500	616.2	1481.9	2181.8	2969.6	3701.6	4434.8	5221.2	6099.6	9622.8	10797.6
5000	624.0	1497.5	2205.1	3000.4	3740.1	4481.1	5275.4	6164.0	9694.0	10877.6
10000	640.5	1528.1	2251.8	3060.1	3816.9	4573.8	5382.9	6279.3	9833.6	11030.2

Table 31: Estimated Running Times (seconds): ISDE_CVKDE, d=10

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	1232.9	3020.6	4485.5	6021.1	7785.8	9245.0	10865.8	12623.1	19982.9	22396.2
2500	1245.5	3045.6	4523.2	6071.0	7847.9	9319.9	10952.6	12722.5	20095.0	22520.9
5000	1261.7	3079.9	4572.8	6141.4	7935.1	9422.7	11072.4	12860.7	20250.8	22686.6
10000	1295.8	3141.8	4666.3	6267.9	8087.7	9615.7	11282.8	13118.9	20524.2	23011.4

Table 32: Estimated Running Times (seconds): ISDE_CVKDE, d=11

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	2502.1	6272.5	9398.4	12377.2	15621.1	18987.9	22390.0	25962.2	41150.6	46385.5
2500	2528.8	6325.4	9476.6	12480.8	15751.7	19144.7	22572.1	26170.4	41385.0	46646.1
5000	2564.3	6401.0	9580.6	12622.9	15927.7	19374.6	22854.7	26479.4	41711.8	47024.8
10000	2632.4	6529.2	9780.8	12892.5	16242.0	19748.8	23280.4	26959.1	42282.9	47645.4

Table 33: Estimated Running Times (seconds): ISDE_CVKDE, d=12

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	5143.0	13041.6	19250.9	25730.1	32419.6	39452.9	46369.7	54352.8	86253.5	95395.9
2500	5197.4	13150.4	19413.6	25949.1	32691.4	39780.0	46751.9	54788.0	86743.6	95940.4
5000	5273.2	13299.5	19645.2	26248.5	33062.3	40262.9	47293.9	55403.6	87448.1	96739.9
10000	5419.9	13575.5	20051.1	26794.1	33765.9	41085.1	48302.8	56511.4	88712.1	98068.1

Table 34: Estimated Running Times (seconds): ISDE_CVKDE, d=13

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	10391.3	26826.3	40281.2	53405.5	67125.1	81904.2	96509.0	111512.3	177267.2	200149.4
2500	10509.0	27057.6	40622.1	53859.8	67694.0	82586.6	97307.8	112418.0	178294.5	201289.4
5000	10666.9	27395.2	41103.2	54518.9	68475.6	83629.5	98500.0	113740.9	179725.4	202942.2
10000	10962.9	27953.0	41901.7	55632.1	69852.4	85224.6	100344.6	116021.9	182182.9	205641.1

Table 35: Estimated Running Times (seconds): ISDE_CVKDE, d=14

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	21392.7	55727.9	84354.7	110258.4	141535.8	168242.8	197592.8	232958.2	371884.2	414921.9
2500	21630.2	56207.9	85072.4	111215.3	142723.9	169663.7	199257.4	234859.4	374018.6	417294.7
5000	21980.8	56923.1	86069.5	112619.2	144488.6	171857.9	201755.8	237397.8	377359.2	420805.1
10000	22560.2	58119.6	87840.1	114899.1	147411.2	175303.3	205721.1	242239.6	382213.1	426463.1

Table 36: Estimated Running Times (seconds): ISDE_CVKDE, d=15

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.6
2500	0.6	0.6	0.6	0.6	0.8	0.6	0.7	0.7	0.7	0.7
5000	0.8	0.8	0.8	0.8	0.8	0.9	1.0	0.9	0.9	1.1
10000	1.1	1.2	1.1	4.1	1.3	1.2	1.3	1.4	1.2	1.4

Table 37: Estimated Running Times (seconds): ISDE Empirical Covariance, d=10

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	0.8	0.9	1.0	0.9	1.0	1.0	1.0	1.3	1.1	1.2
2500	1.2	1.3	1.3	1.3	1.3	1.3	1.9	1.6	1.9	1.5
5000	1.9	2.4	3.1	1.7	2.7	2.4	1.8	3.6	3.4	1.9
10000	7.1	4.3	5.3	5.4	9.5	5.8	2.8	8.3	6.9	2.7

Table 38: Estimated Running Times (seconds): ISDE Empirical Covariance, d=11

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	1.7	1.8	1.9	1.9	2.1	2.0	2.1	2.4	2.3	2.4
2500	2.4	2.5	2.5	2.7	2.8	2.7	2.8	3.0	3.0	3.3
5000	5.6	3.7	5.2	4.1	6.0	4.0	7.6	7.4	3.8	5.3
10000	12.7	6.0	7.1	6.7	8.9	12.6	8.1	8.7	5.2	17.3

Table 39: Estimated Running Times (seconds): ISDE Empirical Covariance, d=12

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	17.9	3.9	15.9	4.0	4.0	4.1	4.5	4.6	4.7	4.8
2500	5.7	14.7	14.8	5.3	5.5	5.6	6.2	5.9	6.1	6.7
5000	7.5	20.4	23.8	7.6	7.2	8.8	7.5	26.2	7.8	9.9
10000	17.4	43.5	23.2	15.8	13.1	17.3	19.8	27.4	24.1	16.1

Table 40: Estimated Running Times (seconds): ISDE Empirical Covariance, d=13

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	7.2	7.2	7.5	8.0	56.1	9.2	9.3	8.3	11.2	51.7
2500	10.2	10.6	13.8	11.3	14.6	14.9	12.6	13.3	15.0	45.2
5000	17.6	35.3	20.8	46.1	41.5	39.6	51.7	15.9	42.4	50.5
10000	27.7	30.8	150.0	135.7	80.7	88.6	47.6	23.6	24.1	107.7

Table 41: Estimated Running Times (seconds): ISDE Empirical Covariance, d=14

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	16.3	17.2	18.7	23.1	17.3	18.5	18.5	19.5	21.1	21.1
2500	23.6	23.2	24.6	33.2	23.6	24.2	27.8	25.0	26.0	26.6
5000	29.2	28.7	20.1	29.3	31.2	29.8	31.4	34.7	32.0	35.0
10000	42.2	44.6	39.1	40.6	41.9	43.2	43.8	46.5	45.5	48.2

Table 42: Estimated Running Times (seconds): ISDE Empirical Covariance, d=15

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	29.6	32.3	32.6	33.1	37.7	36.1	50.1	41.4	42.6	51.4
2500	42.0	42.2	45.2	45.5	47.1	48.2	51.6	57.0	56.6	55.1
5000	53.2	54.9	59.1	87.6	59.5	63.1	70.6	73.3	90.3	71.6
10000	79.0	79.0	84.2	95.9	89.9	91.0	108.5	110.9	106.1	199.1

Table 43: Estimated Running Times (seconds): ISDE Empirical Covariance, d=16

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	58.0	62.7	71.1	117.4	71.5	76.7	80.1	87.3	91.3	96.5
2500	89.4	85.7	93.2	97.1	95.8	98.6	102.3	109.9	117.6	117.3
5000	115.3	114.6	141.8	120.3	124.9	128.0	134.8	140.6	139.1	145.7
10000	158.0	161.9	163.7	174.5	178.8	175.9	179.4	192.6	202.7	201.1

Table 44: Estimated Running Times (seconds): ISDE Empirical Covariance, d=17

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	126.8	128.1	134.7	146.2	154.6	159.3	171.9	184.1	188.9	204.7
2500	176.7	183.9	188.6	197.1	209.2	219.3	215.6	218.9	226.8	260.3
5000	230.7	235.3	248.4	249.6	258.5	287.1	288.1	289.2	295.7	300.7
10000	348.8	334.7	372.1	411.7	384.0	381.4	402.1	397.8	422.4	408.6

Table 45: Estimated Running Times (seconds): ISDE Empirical Covariance, d=18

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	249.4	269.6	288.7	289.0	317.7	329.0	355.4	378.5	391.5	402.7
2500	359.5	372.2	778.4	413.7	411.4	423.0	453.4	454.7	469.4	489.8
5000	469.7	497.0	674.5	527.6	534.6	600.1	574.0	609.9	604.7	603.5
10000	699.8	2180.7	746.3	835.2	782.8	803.8	1119.2	815.2	853.5	879.6

Table 46: Estimated Running Times (seconds): ISDE Empirical Covariance, d=19

n \ m	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
500	497.3	545.1	588.4	638.3	649.6	685.8	707.1	769.1	807.8	844.3
2500	2709.6	776.3	1247.1	905.6	868.9	867.5	934.3	917.3	954.6	1002.9
5000	1897.8	1010.4	1036.1	1041.0	1091.0	1140.9	1165.9	1694.1	1239.6	1263.4
10000	1485.6	1482.4	1513.9	1504.7	1544.1	1642.7	1660.5	1680.0	1729.7	1767.6

Table 47: Estimated Running Times (seconds): ISDE Empirical Covariance, d=20

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	0.90586 ± 1.45386	1.84765 ± 0.7583	3.6083 ± 2.38983	3.42761 ± 1.15221
BDCS	0.33574 ± 0.21546	1.78015 ± 0.58581	2.87573 ± 1.20051	4.4576 ± 1.7183
Empirical Covariance	0.64975 ± 0.27391	3.75359 ± 0.75287	6.8254 ± 1.2289	11.60758 ± 1.74779

Table 48: Gaussian: KL loss ($.10^3$) - $N = 6000$, $\sigma = 0.1$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	0.6496 ± 0.36463	2.10295 ± 0.49494	2.42813 ± 0.93357	3.67274 ± 1.09725
BDCS	0.53416 ± 0.28393	2.13123 ± 0.52902	2.70863 ± 1.21694	3.24106 ± 0.53982
Empirical Covariance	0.85469 ± 0.35287	3.82405 ± 0.70652	6.46502 ± 1.63926	11.0733 ± 1.06413

Table 49: Gaussian: KL loss ($.10^3$) - $N = 6000$, $\sigma = 0.3$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	1.11567 ± 0.56982	1.9215 ± 0.56733	2.79744 ± 0.74082	3.37672 ± 1.00736
BDCS	0.60601 ± 0.2573	1.88644 ± 0.56942	2.73432 ± 1.09249	3.34928 ± 1.60013
Empirical Covariance	1.24109 ± 0.47171	3.28483 ± 0.46321	6.78593 ± 1.06595	10.90951 ± 1.16906

Table 50: Gaussian: KL loss ($.10^3$) - $N = 6000$, $\sigma = 0.5$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	0.62104 ± 0.3457	1.7157 ± 0.52453	3.06233 ± 1.14965	3.49062 ± 0.88227
BDCS	0.4701 ± 0.23901	1.63755 ± 0.67404	2.73209 ± 1.00773	3.96567 ± 1.84841
Empirical Covariance	0.84112 ± 0.25903	3.35138 ± 0.60014	6.55091 ± 1.2976	11.01067 ± 1.77103

Table 51: Gaussian: KL loss ($.10^3$) - $N = 6000$, $\sigma = 0.7$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	0.44956 ± 0.34185	2.18671 ± 0.84376	3.12589 ± 1.0662	4.1686 ± 1.0966
BDCS	0.38017 ± 0.20702	1.88826 ± 0.5567	3.2894 ± 1.42922	2.94753 ± 0.51632
Empirical Covariance	0.62817 ± 0.31976	4.03441 ± 0.95661	7.0306 ± 1.55519	11.33662 ± 0.74408

Table 52: Gaussian: KL loss ($.10^3$) - $N = 6000$, $\sigma = 0.9$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	22.32724 ± 23.06876	60.58244 ± 25.66192	70.66376 ± 11.11799	106.88255 ± 39.7745
BDCS	13.40399 ± 8.77627	64.38877 ± 28.91938	80.27932 ± 26.80539	94.30464 ± 20.77999
Empirical Covariance	27.64968 ± 20.79988	123.24056 ± 34.04314	218.34033 ± 47.68867	384.82419 ± 61.92759

Table 53: Gaussian: KL loss ($.10^3$) - $N = 200$, $\sigma = 0.6$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	6.70797 ± 4.35035	26.92186 ± 11.318	33.62495 ± 9.88967	47.28646 ± 14.996
BDCS	5.93573 ± 3.09991	26.81976 ± 10.81855	38.61735 ± 6.82126	41.68565 ± 15.92671
Empirical Covariance	10.28252 ± 4.66323	48.13902 ± 10.66617	87.41098 ± 12.78745	151.86047 ± 23.27759

Table 54: Gaussian: KL loss ($.10^3$) - $N = 500$, $\sigma = 0.6$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	5.3151 ± 2.48114	12.77626 ± 7.87407	14.35607 ± 2.78512	22.67452 ± 6.34625
BDCS	3.85046 ± 0.84159	10.51985 ± 4.33776	15.448 ± 4.87969	23.98078 ± 6.84809
Empirical Covariance	6.22632 ± 2.01452	21.45073 ± 6.81226	40.26588 ± 6.89861	68.86687 ± 8.71928

Table 55: Gaussian: KL loss ($.10^3$) - $N = 1000$, $\sigma = 0.6$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	1.79894 ± 0.59115	6.48825 ± 1.76735	8.20553 ± 2.52422	9.49403 ± 2.30667
BDCS	1.61396 ± 0.57266	6.17586 ± 2.09449	5.5528 ± 1.72892	9.43337 ± 3.11054
Empirical Covariance	2.58469 ± 0.75941	12.14864 ± 2.54618	18.85504 ± 3.38178	33.87596 ± 2.94877

Table 56: Gaussian: KL loss ($.10^3$) - $N = 2000$, $\sigma = 0.6$

	[2, 2]	[4, 4, 1]	[4, 3, 2, 3]	[4, 4, 3, 3, 2]
ISDE	0.45637 ± 0.15478	1.42778 ± 0.7082	2.08074 ± 0.53012	2.35671 ± 0.4755
BDCS	0.4133 ± 0.13431	1.13536 ± 0.30302	2.01497 ± 0.34142	2.8404 ± 1.23202
Empirical Covariance	0.63192 ± 0.16979	2.64892 ± 0.30215	4.86245 ± 0.72388	8.33952 ± 0.61862

Table 57: Gaussian: KL loss ($.10^3$) - $N = 8000$, $\sigma = 0.6$