# Nonparametric Density Estimation for High-Dimensional Data - Algorithms and Applications

Zhipeng Wang[*] and David W. Scott[†]

**Article Type:**

Advanced Review

## Abstract

Density Estimation is one of the central areas of statistics whose purpose is to estimate the probability density function underlying the observed data. It serves as a building block for many tasks in statistical inference, visualization, and machine learning. Density Estimation is widely adopted in the domain of unsupervised learning especially for the application of clustering. As big data become pervasive in almost every area of data sciences, analyzing high-dimensional data that have many features and variables appears to be a major focus in both academia and industry. High-dimensional data pose challenges not only from the theoretical aspects of statistical inference, but also from the algorithmic/computational considerations of machine learning and data analytics. This paper reviews a collection of selected nonparametric density estimation algorithms for high-dimensional data, some of them are recently published and provide interesting mathematical insights. The important application domain of nonparametric density estimation, such as modal clustering, are also included in this paper. Several research directions related to density estimation and high-dimensional data analysis are suggested by the authors.

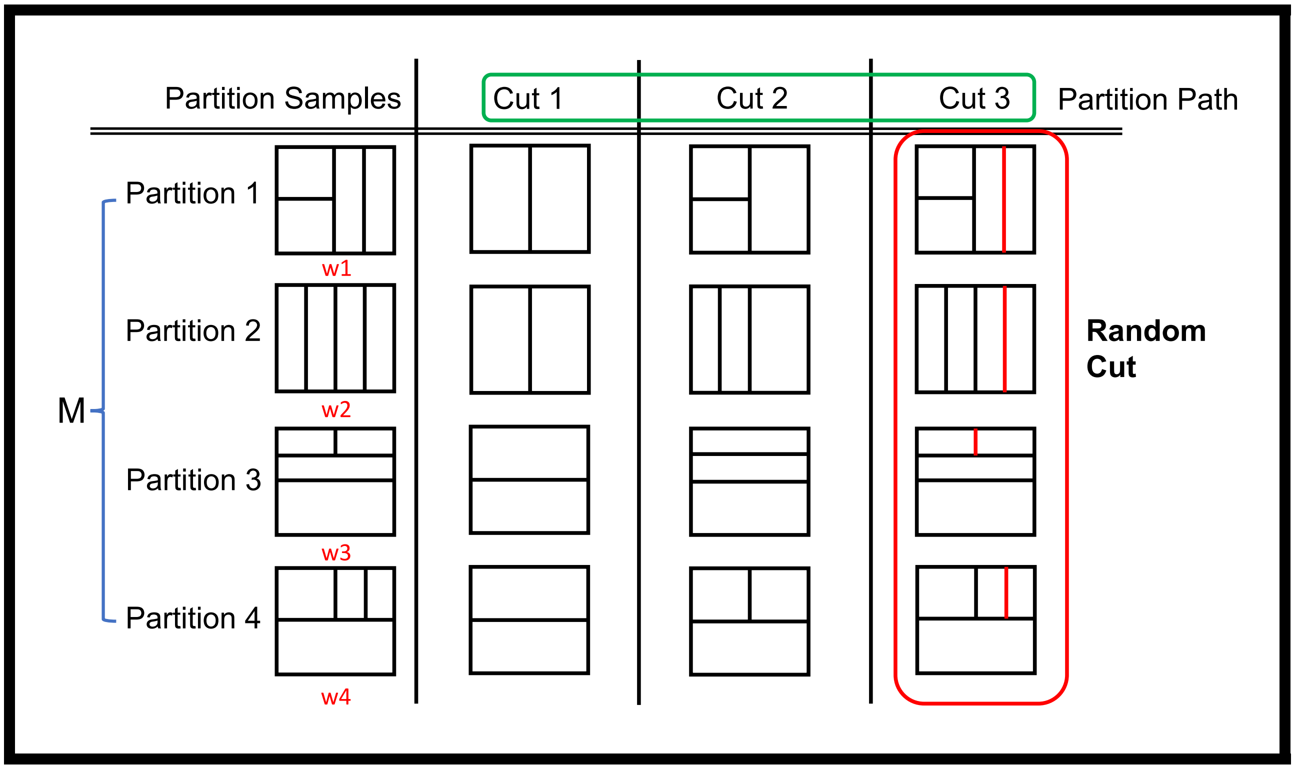
**Keywords:** Density Estimation, High-dimensional data, Clustering, Neural Networks, Data Partitioning

[*]Department of Statistics, Rice University, Houston, TX, email: zw12@rice.edu

[†]Department of Statistics, Rice University, Houston, TX 77251-1892, email: scottdw@rice.edu.

## GRAPHICAL TABLE OF CONTENTS

Include an attractive full color image for the online Table of Contents. It may be a figure or panel from the article, or may be specifically designed as a visual summary. You will need to upload this as a separate file during submission.



# INTRODUCTION

Density Estimation is a widely adopted tool for many tasks in statistical inference, machine learning, visualization, and exploratory data analysis. The aim of density estimation is to approximate the probability density function underlying the data, which are assumed to be $i.i.d.$ Existing density estimation algorithms can be categorized into either parametric, semi-parametric, or nonparametric approaches. Parametric density estimation algorithms are model-based, usually come with strong assumptions on the distribution of the underlying data. One of the most widely-adopted parametric or semi-parametric density estimation algorithms is the Gaussian Mixture Model (GMM), which was first considered by Karl Pearson (1894) [87] followed by many recent works including Aitkin and Wilson (1980) [1], Hathaway (1985) [47], McLachlan and Krishnan (2008) [72], and Wang and Wang (2015) [114]. The basic idea is that given $i.i.d.$ data $\boldsymbol{x}_i \in \Re^d$ , the probability distribution of $\boldsymbol{x}$ can be modeled as a simple linear superposition of Gaussian Components:

$$\hat{f}_K(\mathbf{x}) = \sum_{k=1}^{K} w_k \, \phi(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \,, \tag{1}$$

where the nonnegative weights $w_k$ sum to 1. Choosing the number of Gaussian Component $K$ is usually a tricky task. Theoretically the Gaussian Mixture model can estimate any density function if $K$ is large enough. In practice, however, increasing $K$ would lead to large numbers of parameters to be estimated by the maximum likelihood algorithm. Since $\boldsymbol{x} \in \Re^d$, the most general model contains $K-1$ parameters in the weight vector, $K \times d$ parameters in the mean vectors, and $K \times d(d+1)/2$ parameters in the covariance matrices. This will result in computational challenges and more importantly, will require a much bigger dataset. In the application of clustering, each component in the Gaussian Mixture model naturally corresponds to one cluster, and one expects in the ideal case that the $K$-component Gaussian Mixture Models would illustrate $K$ modes in the density function. Unfortunately, this is not always the case. This problem was pointed out in previous research such as Good and Gaskin (1980) [41] and Roeder (1990) [92]. Another natural challenge is the choice of initial values of the parameters for a maximum likelihood algorithm. If one assumes $\Sigma_k = \Sigma$ for all $k$, then the number of parameters will be significantly reduced at the cost of generality and possibly accuracy, even when $K$ is dramatically increased. One problem is that if we make the assumption of a fully general covariance matrix and if the determinant of any one of the $\Sigma_k$ approaches 0, then the maximum likelihood criterion will approach $+\infty$. However, theoretical results and practical experience show that there are many local maxima in the likelihood function that provide useful estimates (Hathaway, 1985 [47]). Thus trying a number of different initializations is highly recommended. (Anecdotally, the initialization problem is considered to be NP-hard in the machine learning literature.) When applying Gaussian Mixture Modeling, there is considerable interest in the relative size of the components. The estimated weights $\hat{w}_k$ provide a natural choice for this purpose. However, the Gaussian density does not provide an orthonormal basis for density functions. In the $L_2$ function space, the mixture model is dense but the true number of components $K$ might be quite large (or even infinite). However, there are many solutions where the individual components $\{w_k, \mu_k, \Sigma_k\}$ are quite different, but the overall sums in Equation (1) are visually identical [22, 23]. This is the result of the basis not being orthogonal, so that there is high correlation among the estimated parameters in the GMM.

We also note that there is a vast amount of literature on probabilistic graphical models (PGMs). PGMs are a parametric family of statistical models for approximating multivariate joint probability distributions using graphs to express the conditional dependence structure between random variables. They are widely used in probability theory and statistics—especially in Bayesian statistics and machine learning. Useful references are Jordan (2004) [57], Koller (2009) [62], and Wainwright (2010) [112]. Graphical models constitute an active area of research in machine learning and statistics, which we will not cover in this article. Interested readers should refer to the references mentioned above or delve into some relevant extensions and applications; see Wand (2016) [113], Minka (2005) [80], and Yedidia et al. (2003) [119].

The intrinsic problems arising from parametric PDE approaches promote the development of nonparametric density estimation. In this article we will cover some interesting nonparametric density estimation algorithms. Especially we introduce the algorithms that are potentially suited for high-dimensional data. Here we define the ”high-dimensional data” as the data with $3 < d << 50$, where $d$ is the number of dimensions. We understand that it is rather a subjective concept and might have different range given different problems. Our definition of high-dimensional data was motivated by the work of Wong [70, 116] which mentioned that the ideal density estimator should be able to reliably estimate density functions for high-dimensional data with dimensions from 4 to 50. Nonparametric methods provide powerful and flexible means to estimate density functions, and thus have become a very active research topic in the field. Existing nonparametric density estimation algorithms include histograms [95], frequency polygons [96], Kernel Density Estimation [94, 101], Splines [102, 29], and neural network-based density estimation [73, 63, 109, 110, 111, 86]. This field is rapidly developing and new techniques are being created to address the pressing need of big data analytics. They serve as a foundation for many applications such as clustering, which we will also discuss in this paper.

Clustering is one of the most important and challenging problems in machine learning. It may be formulated as an unsupervised learning algorithm in which the class labels are unknown, not even the number of classes $K$. It has wide applications in data compression [21, 76], anomaly detection [66, 52], recommendation systems and Internet of Things (IoTs) [88, 9, 105, 69], etc. Density estimation serves as a foundation for clustering, as one can find modes in the estimated density function, and then associate each mode with a cluster. The modal value itself is taken as the prototypical member of the cluster. The resulting "mode clustering" or "modal clustering" has been extensively studied; see Carmichael et al. (1968) [10] and Hartigan (1975) [45] for seminal works, as well as (Azzalini and Torelli (2007) [3], Cheng (1995) [18], Chazal et al. (2013) [16], Comaniciu and Meer (2002) [24], Fukunaga and Hostetler (1975) [39], Li et al.(2007) [65], Chacón and Duong (2013) [12], Arias-Castro et al.(2013) [2], Minnotte and Scott (1993) [83], Minnotte, Marchette, and Wegman (1998) [82], and Chacón (2016) [13]. The method might generate a conservative result in the sense that pairs of adjacent clusters might manifest as a single mode (or a single bump) in the kernel estimate. But clustering is an exploratory activity, so such limitations should be tolerated. Adding more informative variables might help further separation of the clusters in the high dimensional feature space.

Recent work by Chen (2016) [17] provides several enhancements over the existing mode clustering formulations, including a soft variant of cluster assignments, a measure of connectivity between clusters, a method to denoise small clusters and a way to visualize clusters. A comprehensive survey of modal clustering has recently been provided by Menardi (2016) [79], which should be read in parallel with material below.

In addition to the approaches introduced above, there are many clustering algorithms that do not rely on a parametric or nonparametric probability density estimation of the data. The most commonly used is the hierarchical clustering algorithm, which is implemented based on an iterative distance-based approach; see Johnson (1967) [55] and a recent overview in Izenman (2008) [53]. The results of the algorithm are usually displayed as a binary tree. The most widely used nonhierarchical clustering algorithm is $k$-means (MacQueen (1967) [71]) that iteratively updates the centroids of points currently assigned to the $k$ groups, then reallocates points to the closest centroid, and stops when no further updates occur. Recent work done by Chi and Lange (2015) [20] and Chi et al. (2017) [19] further extended the $k$-means and hierarchical clustering algorithms by proposing splitting algorithms for convex clustering problems. The basic idea is to formulate clustering tasks as a convex optimization problem, in which there is a unique global minimizer for the objective function and the cluster centroids are shrunk toward each other. Then a variety of splitting algorithms such as alternating direction method of multipliers (ADMM) and alternating minimization algorithm (AMA) can be adopted to solve the optimization problem; see Chi et al. (2015) [20].

The remainder of this article will be divided as follow: we will first review some of the important algorithms in nonparametric density estimation, including neural networks-based density estimation algorithms as well as density estimation algorithms based on adaptive data partitioning and Projection Pursuit. Then we will switch our focus to mode clustering methods using nonparametric density estimation. Finally, we will provide critical comments on the limitations of density-based algorithms and suggest future research directions.

## NONPARAMETRIC DENSITY ESTIMATION FOR HIGH-DIMENSIONAL DATA

In the following sections, we will introduce relevant nonparametric algorithms for high-dimensional density estimation. Since neural networks gained popularity in recent years, we want to cover some relevant density estimation algorithms based on neural networks. We will also introduce algorithms of Multivariate Density Estimation via adaptive sequential data partitioning, which were proposed by Luo and Wong (2013) [70] and Li and Wong (2016)[64]. These density estimation algorithms provide both computationally efficient and statistically robust means for function estimation. Projection Pursuit Density Estimation (PPDE), which was first introduced by Friedman and Tukey [38] has evolved into an efficient method of density estimation for high-dimensional data. We will also discuss the PPDE algorithm in this section.

## Density Estimation Based on Neural Networks

One of the problems in kernel density estimation is that small changes of data and smoothing parameters can lead to large fluctuations in the estimated density. In order to make the estimation more robust to the slight changes of data, some regularization is usually needed. The regularizations are often reflected by choosing the smoothing parameters (kernel width or number of kernel functions $K(\cdot)$). However, the estimated density will be extremely sensitive to the choice of the smoothing parameter. A poor choice can lead to either oversmoothing or undersmoothing, either globally, locally, or both.

The method of Neural Networks has recently gained tremendous popularity in the machine learning community. It provides a powerful capability to estimation any function to any given precision while maintaining the flexibility to choose an error function to fit into the application. Neural network consists of many interconnected neurons, each neuron performs a nonlinear feature mapping $\sigma(W^T\boldsymbol{x} + b)$, where $\boldsymbol{x} \in \Re^d$ is the input data, $W$ is the weight vector for the neuron, and $\sigma$ is the nonlinear function (which is usually implemented as either sigmoid or ReLU function in practice [42]). The underlying intuition is that the neural network can somehow learn the abstract representation of data by the exhaustive nonlinear mapping of the original features. Density estimation using neural networks once was used very sporadically due to the limitation of computing resources. Magdon-Ismail and Atiya (1998) [73] proposed two methods of density estimation that can be implemented using multilayer neural networks. One is the stochastic learning of cumulative distribution function, which only works for univariate density estimation.

Let $x_n \in \Re^1$, where $n = 1, ..., N$ and the underlying density is $g(x)$. Its cumulative distribution function is $G(x) = \int_{-\infty}^{x} g(x')\, dx'$. The density of the random variable $G(X)$ is uniform in $[0, 1]$. Let the network output be denoted by $H(x, w)$. The aim is to have $H(x, w) = G(x)$. The basic idea behind the algorithm is to use the $N$ original data points as the input, and at each iteration cycle, new data points that are generated from a uniform distribution on the interval of $[0, 1]$ as the network targets. The weights are then adjusted to map the new data points. Thus the neural network is trained to map the data to a uniform distribution. The algorithm is illustrated as follows:

1. Let $x_1 \le x_2 \le ... \le x_N$ be the data points. Set the iteration number of the training cycle $t = 1$. Initialize the weights of the neural network randomly to $w_1$.

2. Randomly generate N data points from a uniform distribution in $[0, 1]$, and sort them in ascending order $u_1 \le u_2 \le ... \le u_N$. Those points $u_n$ are the target output for the neural network with input $x_n$

3. Update the network weights according to the backpropagation scheme:

$$w_{t+1} = w_t - \eta_t \frac{\partial J}{\partial w} ,$$

   where $J$ is the objective function, and $\eta_t$ is the learning rate at each iteration. The objective function $J$ includes the error term and the monotonicity penalty term:

$$J = \sum_{n=1}^{N} \big[H(x_n, w) - u_n\big]^2 + \lambda \sum_{k=1}^{N_h} \Theta\big(H(y_k, w) - H(y_k + \Delta, w)\big)\big[H(y_k, w) - H(y_k + \Delta, w)\big]^2 .$$

   The first term is the standard L-2 error term, and second term is the monotonicity penalty term, $\lambda$ is a positive weighting constant, $\Delta$ is a small positive number, $\Theta(x)$ is the familiar unit step function, and the $y_k$ are any set of data points where the monotonicity is enforced.

4. Move to the next iteration cycle $t = t + 1$ , and go to step 3. Repeat the process until the error is small enough. The resulting density estimate is the derivative of $H$.

Another method they introduced is the smooth interpolation of the cumulative distribution (SIC), which works for multivariate density estimations. The basic idea is that given the input data point $\boldsymbol{x} \in \Re^d$, if the ground truth density function is $g(\boldsymbol{x})$, then the network target output is the corresponding cumulative distribution $\hat{G}(\boldsymbol{x})$. Let $\boldsymbol{x} = (x^1, ..., x^d)^T$, $G(\boldsymbol{x})$ is given by:

$$G(\boldsymbol{x}) = \int_{-\infty}^{x^1} ... \int_{-\infty}^{x^d} g(\boldsymbol{x})\, dx^1 \dots x^d \,. \tag{2}$$

Then we can approximate $G(\boldsymbol{x})$ by using the fraction of data points falling in the area of integration:

$$\hat{G}(\boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^{N} \Theta(\boldsymbol{x} - \boldsymbol{x}_n) \tag{3}$$

where $\Theta$ is defined as:

$$\Theta(\boldsymbol{x}) = \begin{cases} 1 & x^i \geq 0 \;\; (i = 1, 2, ..., d) \\ 0 & \text{otherwise} \,. \end{cases} \tag{4}$$

The $\hat{G}(\boldsymbol{x})$ is an estimate of $G(\boldsymbol{x})$ that is used for the target outputs of the neural network. The neural network model provides a smooth interpolation of the cumulative distribution function which is highly desirable. The density function is then obtained by differentiation of the network outputs with respect to its inputs.

For low-dimensional problems, we can do uniform sampling in (3) using a grid to empirically obtain examples for the target output of the network. For high-dimensional problems beyond two or three dimensions, the uniform sampling becomes computationally expensive. The alternative option is to use the input data points to form examples. To illustrate this, the target output for a input point $\boldsymbol{x}_m$ would be:

$$\hat{G}(\boldsymbol{x}_m) = \frac{1}{N-1} \sum_{n=1, n \neq m}^{N} \Theta(\boldsymbol{x}_m - \boldsymbol{x}_n) \,. \tag{5}$$

Finally the monotonicity of the cumulative distribution function can be used as a hint to guide the training process. The network output $H(\boldsymbol{x}, w)$ approximates the cumulative distribution function $G(\boldsymbol{x})$, then the density estimate can be derived as:

$$\hat{g}(\boldsymbol{x}) = \frac{\partial^d H(\boldsymbol{x}, w)}{\partial x^1 \dots \partial x^d} \,. \tag{6}$$

There are a variety of choices for the neural network architecture. Feedforward neural network is commonly adopted (including both single and multiple hidden layers) [73]; however, it suffers from a number of problems such as gradient vanishing, overfitting and curse of dimensionality. Some regularization techniques such as dropout are commonly used to tackle those problems. There are also other types of architectures such as convolutional neural networks (CNNs), attention-based CNNs, Variational Autoencoder (VAE), Restricted Boltzmann Machines (RBMs) etc. that have much better performance for high-dimensional data. We review some more extended work on using more sophisticated neural network architectures for density estimation.

The early work using neural networks to perform density estimations was extended by Iain Murray and his colleagues [63, 109, 110, 111], Bengio and Bengio (2000) [6], and Gregor and LeCun (2011) [44]. Their approaches combine probabilistic graphical models (either directed or undirected) with neural networks such as restricted Boltzmann machines (RBMs) and feed-forward neural networks. Among the seminal ideas in their work is the Neural Autoregressive Density Estimator (NADE) [63, 111], which starts by factoring any $d$-dimensional distribution $p(\boldsymbol{x})$ $(\boldsymbol{x} \in \Re^d)$ into conditional probabilities (for simplicity $\boldsymbol{x}$ is assumed to be a binary vector):

$$p(\boldsymbol{x}) = \prod_{k=1}^{d} p(x_k \mid \boldsymbol{x}_{s<k}) \,, \tag{7}$$

where $\boldsymbol{x}_{s<k}$ is the first $k-1$ subvector of the vector $\boldsymbol{x}$. The autoregressive generative model of the data is defined by parameterizations of the $d$ conditional distributions $p(x_k \mid \boldsymbol{x}_{s<k})$. Frey et al. (1996) [34] modeled the conditionals via log-linear logistic regressions, which yielded a competitive result. Bengio and Bengio (2000) [6] extended the approach by modeling the conditionals via single-layer feed-forward neural networks. This gained some improvement in model performance at the cost of very large model complexity for high-dimensional datasets. In NADE [111], they also model the conditionals using feed-forward neural networks via the following parameterizations:

$$p(x_k = 1 \mid \boldsymbol{x}_{s<k}) = \sigma(V_k \cdot \mathbf{h}_k + b_k) \tag{8}$$

$$\mathbf{h}_k = \sigma(W_{s<k} \cdot \boldsymbol{x}_{s<k} + c)\,, \tag{9}$$

where $h$ is the hidden unit and $H$ is the number of hidden units. Then $W \in \Re^{H\times d}$ is the weight matrix for hidden units, $V^{d\times H}$, $b \in \Re^d$, and $c \in \Re^H$ are parameters associated with NADE models. Here $\sigma(\theta) = \frac{1}{1+e^{-\theta}}$ is the sigmoid activation function. The weight matrix $W$ and bias $c$ are shared among all the hidden layers $\mathbf{h}_k$ having the same size (shown in Figure 1). This will reduce the total number of parameters from $O(Hd^2)$ to $O(Hd)$. Training the NADE can be done via maximum likelihood, or one can simply minimize the average negative log-likelihood:

$$-\frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{x}^{(n)}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{d}\log p(x_k^{(n)} \mid \boldsymbol{x}_{s<k}^{(n)})\,, \tag{10}$$

where $N$ is the number of training samples. Minimization of the objective function shown above can be readily achieved by stochastic (batch) gradient descent. Since there are $O(Hd)$ parameters in NADE, calculating $p(\boldsymbol{x})$ costs only $O(Hd)$, so the gradient of log-likelihood of training samples can also be calculated with the complexity $O(Hd)$.

The algorithms for calculating $p(\boldsymbol{x})$ and $-\nabla \log p(\boldsymbol{x})$ in Uria et al. (2016) [111] are illustrated in Algorithm 1.

**Algorithm 1:** Computation of $p(\boldsymbol{x})$ and learning gradient in NADE (Uria et al. 2016) [111]

```
**Input:** training sample vector $\boldsymbol{x}$ and ordering $s$ of the input dimensions
**Output:** $p(\boldsymbol{x})$ and gradients $-\nabla \log p(\boldsymbol{x})$ on parameters
**Computation** of $p(\boldsymbol{x})$
Set $\theta_1 \leftarrow c$
Set $p(\boldsymbol{x}) \leftarrow 1$
**For** $k$ from $1$ to $d$ **do**
    Set $\mathbf{h}_k \leftarrow \sigma(\theta_k)$
    $p(x_k = 1 \mid \boldsymbol{x}_{s<k}) \leftarrow \sigma(\mathbf{V}_k \cdot \mathbf{h}_k + b_k)$
    $p(\boldsymbol{x}) \leftarrow p(\boldsymbol{x})(p(x_k = 1 \mid \boldsymbol{x}_{s<k})^{x_k} + (1 - p(x_k = 1 \mid \boldsymbol{x}_{s<k}))^{1-x_k})$
    $\theta_{k+1} \leftarrow \theta_k + \mathbf{W}_k \cdot x_k$
**end for**

**Computation** of learning gradients of $-\log p(\boldsymbol{x})$
Set $\delta\theta_D \leftarrow 0$
Set $\delta c \leftarrow 0$
**for** $k$ from $d$ to $1$ **do**
    $\delta b_k \leftarrow (p(x_k = 1 \mid \boldsymbol{x}_{s<k}) - x_k)$
    $\delta \mathbf{V}_{s<k} \leftarrow (p(x_k = 1 \mid \boldsymbol{x}_{s<k}) - x_k)\mathbf{h}_k^\top$
    $\delta \mathbf{h}_k \leftarrow (p(x_k = 1 \mid \boldsymbol{x}_{s<k}) - x_k)\mathbf{V}_k^\top$
    $\delta c \leftarrow \delta c + \delta \mathbf{h}_k \odot \mathbf{h}_k \odot (1 - \mathbf{h}_k)$
    $\delta W_k \leftarrow \delta\theta_k x_k$
    $\delta\theta_{k-1} \leftarrow \delta\theta_k + \delta \mathbf{h}_k \odot \mathbf{h}_k \odot (1 - \mathbf{h}_k)$
**end for**
**return** $p(\mathbf{x})$, $\delta\mathbf{b}$, $\delta\mathbf{V}$, $\delta\mathbf{c}$, $\delta\mathbf{W}$
```

In their earlier work, Larochelle and Murray (2011 [63]) discussed the relationship between NADE and Restricted Boltzmann Machines (RBMs). In RBMs, it is often intractable to compute the high-dimensional probability distribution because the partition function is intractable, even when the number of hidden units is only moderately large. NADE approximates the partition function using mean field variational inference that makes the probability distribution completely tractable to compute. Uria et al. (2013, 2016) [109, 111] also extended NADE for real-valued variables. Interested readers should refer to the corresponding references for details.

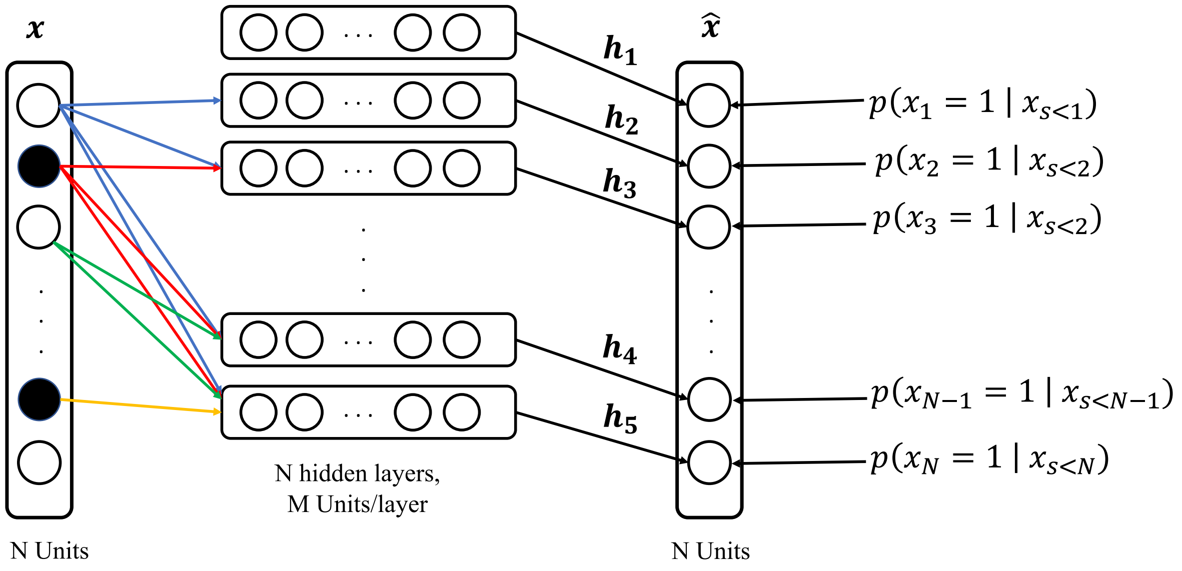

$\boldsymbol{x}$
$\hat{\boldsymbol{x}}$
$h_1$
$h_2$
$h_3$
$h_4$
$h_5$
$p(x_1 = 1 \mid x_{s<1})$
$p(x_2 = 1 \mid x_{s<2})$
$p(x_3 = 1 \mid x_{s<2})$
$p(x_{N-1} = 1 \mid x_{s<N-1})$
$p(x_N = 1 \mid x_{s<N})$
N hidden layers,
M Units/layer
N Units
N Units

Figure 1: The architecture of a Neural Autoregressive Density Estimation (NADE) model. The input vector $\boldsymbol{x}$ is a N-dimensional binary vector, units with value 0 are shown in color black, while the units with value 1 are shown in color white. N input units represents the N dimensions in vector $\boldsymbol{x}_o$. We basically model each conditional probability density $p(x_d = 1 \mid x_{s<d})$ using a single layer feed-forward neural network. There are N hidden layers to model N conditional probabilities. $\mathbf{h_d}$ represents the d-th hidden layer ($d = 1, ..., N$). The output of each hidden layer is calculated via Equation 9. In this example, the vector $\hat{\boldsymbol{x}}$ represents the output, and its dimensions $\hat{\boldsymbol{x}}_i$ ($i = 1, .., N$) are the output of corresponding hidden layer $\mathbf{h_i}$. Notice that each input unit connecting to the hidden layer through the weight-sharing scheme, which is highlighted in the figure with the same color. [111]

One of the limitations in NADE comes from its underlying assumption that the joint density function can be factorized into sequential conditional densities. In many real-world scenarios, the copula models of joint probability should be adopted. Interested readers can refer to Liu (2012)

[67] and Dobra (2011) [28].

Another critical drawback of NADE is that it is sensitive to the sequential order of the variables. For example, given $\boldsymbol{x} \in \Re^d$, and let $\Pi_i(\boldsymbol{x}), i = 1, ..., N$ be a permutation order among elements $(x_1, ..., x_d)$ in $\boldsymbol{x}$. A model with $\Pi_i(\boldsymbol{x})$ and $\Pi_j(\boldsymbol{x}), j \neq i, j \in \{1, ..., N\}$ will likely to have different capability to learn certain density functions. In practice, it is difficult to know which particular sequential order of the variables for the conditional factorization is optimal for the task. One solution to this problem is to train NADE with an sequential order of the variables at random, and combine the predictions from different sequential orders to form an ensemble model [40, 110]. This requires $d$ sequential computations for estimating the density $p(\boldsymbol{x})$ because a hidden state needs to be updated sequentially for every variable. The computational disadvantage of the straightforward solution is not well-suited for large-scale parallel computation. Papamakarios and Murray (2017) [86] recently proposed a method called Masked Autoregressive Flow (MAF) that enables different sequential order of the variables in the conditional factorial and is well-suited for large parallel architecture such as GPUs. The proposed method can also perform density function estimation for real-valued variables.

Given an autoregressive model as follows:

$$p(\boldsymbol{x}) = \prod_{k=1}^{d} p(x_k \mid \boldsymbol{x}_{s<k}) \,, \tag{11}$$

each of the conditionals can be modeled as a single Gaussian distribution. To illustrate this, the $k^{th}$ conditional factor is given as follows:

$$p(x_k \mid \boldsymbol{x}_{s<k}) = \mathcal{N}\Big(x_k \mid \mu_k, (\exp(\beta_k))^2\Big) \,, \tag{12}$$

where $\mu_k = f_{\mu_k}(\boldsymbol{x}_{s<k})$ and $\beta_k = f_{\beta_k}(\boldsymbol{x}_{s<k})$. Note $f_{\mu_k}$ and $f_{\beta_k}$ are real-valued scalar functions that compute the mean and log standard deviation of the $i^{th}$ conditional distribution given all the "previous" variables. The model also uses the vector of random variables $\mathbf{u} = (u_1, ..., u_d)$ to generate data through the following recursive steps:

$$x_k = u_k \exp(\beta_k) + \mu_k \, , \tag{13}$$

where $\mu_k = f_{\mu_k}(\boldsymbol{x}_{s<k}), \beta_k = f_{\beta_k}(\boldsymbol{x}_{s<k})$, and $u_k \sim \mathcal{N}(0,1)$.

The MAF model stems from Normalizing flows [91], which expresses the joint density function $p(\boldsymbol{x})$ through the invertible, differentiable function $f$ of a low-level density $q_u(\mathbf{u})$. It is straightforward to see that $\boldsymbol{x} = f(\mathbf{u})$ where $\mathbf{u} \sim q_u(\mathbf{u})$. The density $q_u(\mathbf{u})$ should be carefully chosen so that it is easy to be evaluated at any variable value of $\mathbf{u}$ (e.g. standard Gaussian). Under the theorem of invertible functions, the joint density $p(\boldsymbol{x})$ can be expressed as:

$$p(\boldsymbol{x}) = q_u\big(f^{-1}(\boldsymbol{x})\big) \cdot \left|\det(\frac{\partial f^{-1}}{\partial \boldsymbol{x}})\right| . \tag{14}$$

In order to compute the density $p(\boldsymbol{x})$, the function $f$ has to be easily invertible, and the determinant of the Jacobian should be easy to compute. Go back to the MAF, $\boldsymbol{x} = f(\mathbf{u})$, where $\mathbf{u} \sim \mathcal{N}(0, \mathbf{I})$. Then given a data point $\boldsymbol{x} \in \Re^d$, the random number $\mathbf{u}$ will be derived from the following steps:

$$u_k = (x_k - \mu_k)\exp(-\beta_k), \;\; \mu_k = f_{\mu_k}(\boldsymbol{x}_{s<k}), \;\; \beta_k = f_{\beta_k}(\boldsymbol{x}_{s<k}) \, . \tag{15}$$

In the Autoregressive model, the Jacobian of $f^{-1}$ has a triangular structure, so the determinant is:

$$\left|\det(\frac{\partial f^{-1}}{\partial \boldsymbol{x}})\right| = \exp\left(\sum_k \beta_k\right) \, , \tag{16}$$

where $\beta_k = f_{\beta_k}(\boldsymbol{x}_{s<k})$.

The density $p(\boldsymbol{x})$ can be obtained by substituting Equations (15) and (16) into Equation (14), so it can also be interpreted as a normalizing flow [58]. The implementation of the set of functions $\{f_{\mu_k}, f_{\beta_k}\}$ with masking borrows the idea from the Masked Autoencoder Density Estimation (MADE) [40]. MADE is simply a feed-forward neural network which takes the input data $\boldsymbol{x}$ and outputs mean $\mu_k$ and variance parameter $\beta_k$ for all $k$ with a single round of pass. In MAF, the weight matrices of MADE are multiplied by the binary masks to ensure that the autoregressive properties are well-maintained. In other words, MAF uses the MADE with Gaussian conditionals as the building layer of the flow. The flow in MAF is interpreted as a flow of autoregressive models through stacking multiple autoregressive model instances, which improves the model fit. The reason to use the masking approach is that it enables transforming the input data $\boldsymbol{x}$ into the random number $\mathbf{u}$ and calculating the density $p(\boldsymbol{x})$ by finishing one round pass through the flow, instead of doing recursive calculations as in Equation (15).

MAF can potentially learn complex density functions for high-dimensional data. Using images from MNIST datasets as examples (note that MNIST datasets stand for Modified National Institute of Standards and Technology datasets, which is a large database of handwritten digits widely used in training many machine learning algorithms for image processing), MAF can successfully learn the complex density functions the MNIST images, and generate images which capture the underlying patterns of the real images; see Figure 2. However, compared to modern image-based generative models such as PixelCNN++ [93], RealNVP [7] or CycleGAN [120], the MAF-generated images lack the fidelity provided by those models. But MAF was originally designed as a general-purpose density estimator rather than a domain-specific generative model. Interested readers can refer to those original papers for details.

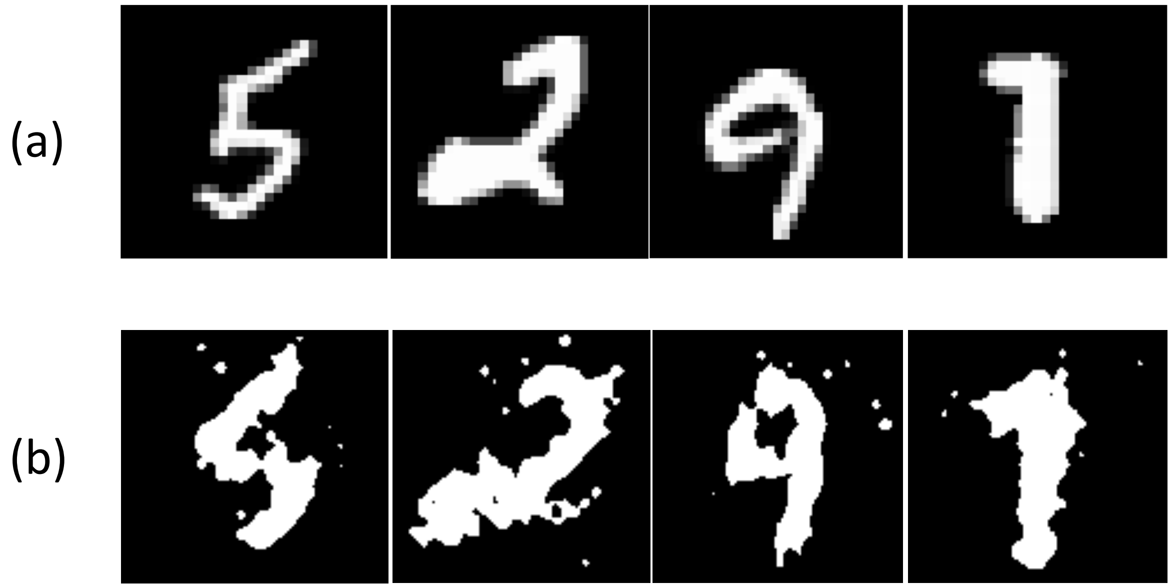


Figure 2: Real Images and Generated Images by MAF from MNIST datasets. (a) Real Images from MNIST datasets; (b) Generated Images by MAF through MNIST datasets. [86]

## Density Estimation Based on Adaptive Data partitioning

It is widely accepted that kernel-based nonparametric density estimation is computationally intense, and it also suffers from the problem of "curse of dimensionality." Essentially as the dimension increases, the number of data points needed to get a reliable density estimator grows exponentially. How can we come up with a reliable density estimator for high-dimensional data with limited amounts of data and computational resources?

Wong and his colleagues have come up with some novel methods to address this problem (Wong and Ma, 2010; Lu and Wong, 2013; Li and Wong, 2016) [117, 70, 64]. The basic idea is to treat the multivariate density estimation as a nonparametric Bayesian problem. Suppose $X$ is a random variable on a space $\Omega$ ($X \in \Re^d$), their distribution $Q$ is unknown but it is assumed to be drawn from a prior distribution $\pi$. The posterior $P(Q \mid X)$ can be calculated by $\pi(Q)Q(X)$. Choosing prior distributions should follow Ferguson's criteria that 1) there should be a large support for the prior and 2) the posterior distribution should be tractable. Although the commonly-adopted Dirichlet process satisfies the Ferguson's criteria, the corresponding posterior does not possess a density function. So instead they considered a class of piecewise constant density functions over partitions of the data space: $Q(X) = \sum_{i=1}^{m} c_i I_{A_i}(X)$, where $m$ is the number of partitions, $A_i$ is the $i$th partition, and $I_{A_i}$ is the indicator function. As usual, $I_{A_i}(X) = 1$ indicates that $X$ falls into the $i$th partition; otherwise $I_{A_i}(X) = 0$. From the piecewise constant prior distribution over partitions, they derived a closed-form marginal posterior distributions for corresponding partitions. The inference on the partitions is achieved by their proposed algorithm that they called "Bayesian sequential partitioning (BSP)." The basic idea is that in each iteration, binary partitioning on one of the subregions and dimensions of the data domain $\Re^d$ is performed. Then the posterior distribution of the corresponding partitions, $\pi^0(\cdot)$, is calculated and used to assign "scores" to each partition. Since the closed-form posterior distribution over partitions is available, the inference of partitions can be done via Markov Chain Monte Carlo (MCMC) or Sequential Importance Sampling (SIS).

Figure 3 illustrates the path for BSP while Figure 4 shows how the Sequential Importance Sampling works. In the space of partitioning paths, the density is defined as $q(g_t)$, which is proportional to the posterior probability for the partition generated from the partitioning path $g_t = (d_1, d_2, ..., d_t)$, where each $d_j, j \in \{1, 2, ..., t\}$ represents the partition decision at level $j$ for dividing a subregion in the partition generated by $g_{t-1} = (d_1, d_2, ..., d_{t-1})$. Let $x_t$ denote the partition generated from $g_t$. Since there are potentially many partition paths that lead to the same partition $x_t$, the author introduced the notation $\Lambda(x_t)$ to present the number of unique paths that lead to the same partition $x_t$. If the partition paths $g_t^{(1)}, g_t^{(2)}, ..., g_t^{(m)}$ are generated from the probability distribution of paths $q(\cdot)$, the corresponding partitions are $x_t^{(1)}, x_t^{(2)}, ..., x_t^{(m)}$. Here, each $x_t^{(i)}$ is the partition generated from the path $g_t^{(i)}$, where $x_t^{(i)} = x(g_t^{(i)})$, can be treated as a weighted sample with the posterior distribution $\pi^0(g_t)$ for the partition and the weights $w_i = 1/\Lambda(x_t^{(i)}), i \in \{1, 2, ..., m\}$. Since $q(g_t) \propto \pi^0(x(g_t))$ and the partition path is sequentially constructed, the weighted samples of partition paths can be generated by using Sequential Importance Sampling [43, 59, 68] as shown in Figure 4.

They calculated the partition score, which is just the logarithm of the posterior probability (as a function of the number of partitions $t$) and the KL divergence between the estimated density and the true density as a function of $t$ for a simulated mixture Gaussian Distribution. Their results indicate that the partition score tracks the KL divergence with higher partition scores corresponding to smaller KL divergences; see Wong [116].

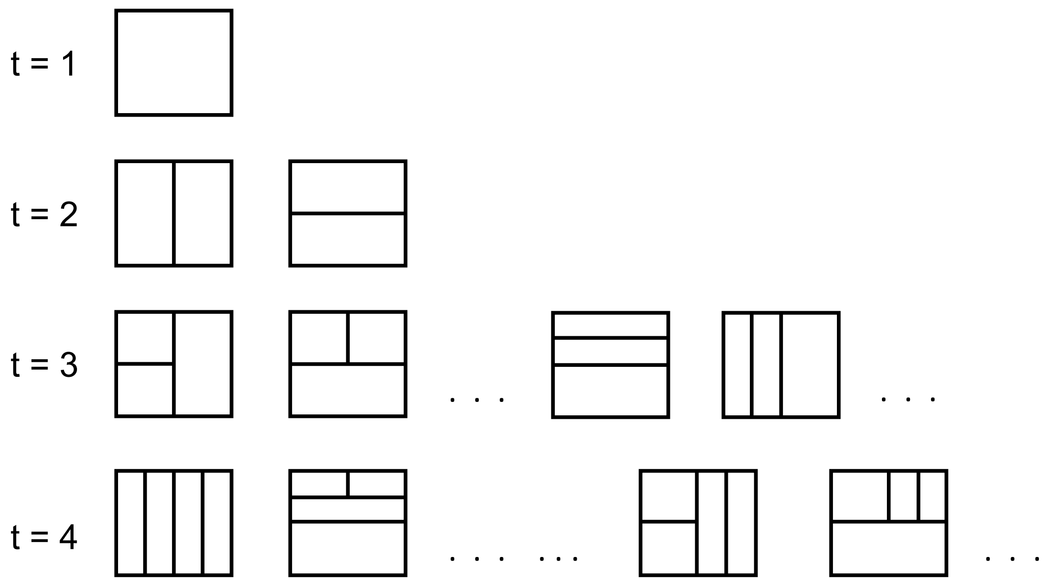


Figure 3: Recursive Sequential Binary Partitioning, where $t = 1, 2, 3, 4$ represent the level of partition, and the partition is performed sequentially. At each level, there are a variety of different ways to perform binary partition; (from Wong (2014) [116]).

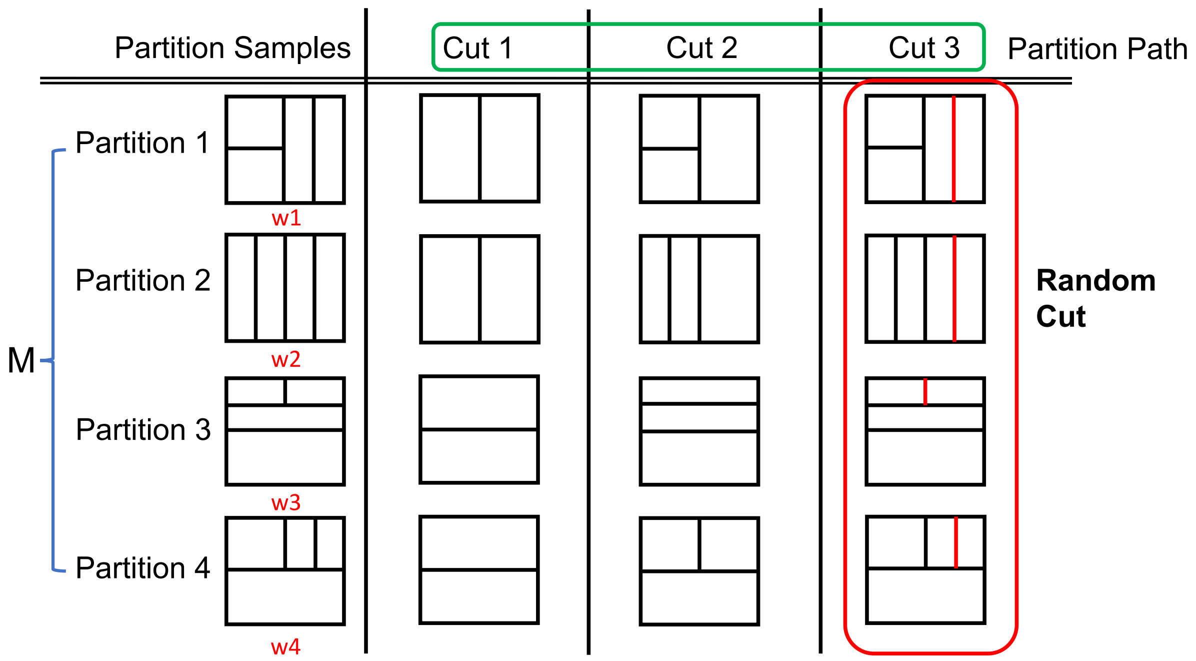


Figure 4: Sequential Importance Sampling (SIS) to generate weighted samples of partition paths. Here 4 partition samples are illustrated, and their corresponding weights are $w_1, w_2, w_3, w_4$, respectively; (from Wong (2014) [116]).

Li et al. (2016) [64] further extended the BSP method, leading to a more flexible and intuitive algorithm for multivariate density estimation. They use a greedy algorithm to determine the splitting point by picking the maximum gap, $g_{jk}$, where $j$ represents $j$th dimension and $k$ represents the $k$th splitting point along any of the dimensions (any dimension will be divided into $m$ equally-spaced bins). Given $n$ data points $X_n = x_1, x_2, ..., x_n$ transformed into $[0,1]^d$, $g_{jk} = \left| (\frac{1}{n}) \sum_{i=1}^{n} \mathbf{1}(x_{ij} < a_j + (b_j - a_j)k/m) - k/m \right|$, for $k = 1, 2, ..., m-1$. There are $(m-1)d$ recorded in total. At each iteration the maximum gap will be picked and the corresponding splitting point will be selected. The algorithm keeps iterating until the maximum gap falls below a predetermined threshold. The algorithm is computationally efficient and has been proven to converge. For more technical details readers are encouraged to look into the original reference.

There are several potential limitations of BSP algorithm and its extended version: 1) data partitioning has to be sequentially performed, so it might take a long time to converge when number of dimensions gets high; 2) the resulting piecewise constant density function is discontinuous, in which case the edge problem from multivariate histogram estimators will arise, so the density estimation might be rather biased; and 3) for the BSP, it is not always appropriate to choose a prior distribution that is piecewise constant, so that different priors might be needed, in which case the posterior distribution might not possess a closed-form expression. One potential way to improve the statistical bias is by using a multivariate frequency polygon (Hjort, [50]). Another potential way is to identify clusters via spectral clustering [99, 78, 27], and then perform discrete convolution over the entire data domain, which is an ongoing project by the current authors and will not be covered by this article.

## Projection Pursuit Density Estimation (PPDE)

One of the first "advanced" algorithms for finding structure (read: clusters) in data was devised by Friedman and Tukey [38]. A nonparametric criterion was presented that used to find low-dimensional projections with "interesting" views. They estimated that interactive examination of all projections would quickly become inefficient for a human, and so they devised "cognostics" or criteria that would allow the computer to perform an extensive search without human intervention. Interesting projections (local optima in the cognostics space) could then be presented to the human user. They called their algorithm *projection pursuit* (see also [35]).

A number of extensions have been investigated subsequently, namely, projection pursuit regression [36] and projection pursuit density estimation (PPDE) [37]. The motivation behind the PPDE algorithm is to tackle the poor performance of multivariate kernel density estimation when it comes to high-dimensional data, because extremely large sample sizes are needed to achieve the level of numerical accuracy available in low dimensions. The PPDE algorithm uses an iterative procedure to find the interesting subspace, which is spanned by a few significant components. The detailed procedure is outlined in Algorithm 2.

**Algorithm 2:** Projection Pursuit Density Estimation (PPDE) Algorithm

```
Input: Observed Data, $L = \{x_i, i = 1, 2, ..., n\}, x_i \in \Re^d$, Scale the data
to have mean $\mathbf{0}$ and covariance matrix $\mathbf{I}_d$
Initialization: Choose $\hat{p}^{(0)}$ to be an initial density estimate of $p(\boldsymbol{x})$,
                usually it is taken to be standard multivariate Gaussian
For $k = 1, 2, ...$
    1. Find the direction $\mathbf{c}_k \in \Re^d$ for which the marginal (model) $p_{\mathbf{c}_k}(\boldsymbol{x})$ along
       $\mathbf{c}_k$ differs most from the current estimated data marginal $\hat{p}_{\mathbf{c}_k}^{(k-1)}$ along $\mathbf{c}_k$
       the choice of $\mathbf{c}_k$ generally will not be unique)
    2. Given $\mathbf{c}_k$, calculate the univariate "ridge" function:
       $t_k(\mathbf{c}_k^\tau \boldsymbol{x}) = \frac{p_{\mathbf{c}_k}(\mathbf{c}_k^\tau \boldsymbol{x})}{\hat{p}_{\mathbf{c}_k}(\mathbf{c}_k^\tau \boldsymbol{x})}$
    3. Update the previous estimate of the probability density function:
       $\hat{p}^{(k)}(\boldsymbol{x}) = \hat{p}^{(k-1)}(\boldsymbol{x}) t_k(\mathbf{c}_k^\tau \boldsymbol{x})$
End For
```

Just to clarify the notation used in Algorithm 2, the vectors $\{\mathbf{c}_k\}$ are unit-length directions in $\Re^r$, and the ridge functions $\{t_k\}$ are constructed so that $\hat{p}^{(k)}$ converges to $p$ numerically as $k \to \infty$. The number of iterations $k$ serves as a smoothing parameter, and the iteration ceases when the stopping rule determines that the estimation bias is balanced out against the estimation variance. Computation of the ridge function $t_k(\mathbf{c}_k^\tau \boldsymbol{x})$ can be done via two steps: 1) given $\mathbf{c}_k$, project the sample data along the direction $\mathbf{c}_k$, thus obtaining $q_i = \mathbf{c}_k^\tau x_i, i = 1, 2, ..., n$; and 2) compute a kernel density estimate from the projected data $\{q_i\}$. Computing the $\hat{p}_{\mathbf{c}_k}$ is done via Monte Carlo sampling followed by a kernel density estimation. Alternative smoothing methods include cubic spline functions [37] and the average shifted histograms [54].

The specific use of projection pursuit for finding clusters has been investigated recently by Tasoulis, et al. [107]. However, the underlying technology has not seen the rapid development seen by algorithms and has been limited to projection to a few dimensions. Given that projection tends to reduce the number of modes (clusters) by combination (overlap), we do not pursue this further. A nice review is provided by Jones and Sibson [56].

## APPLICATIONS of NONPARAMETRIC DENSITY ESTIMATION: MODAL CLUSTERING

In this section, we would like to discuss one of the important application domains of nonparametric density estimation, which is modal clustering. Modal clustering is a clustering approach that determines the clusters of data through identifying modes in the probability density function. Each mode is then associated with a cluster. The technical challenge is to discover all the true modes in the density function through data-driven approach. Good and Gaskin (1980) [41] pioneered the way of using nonparametric density estimations as a powerful tool for discovering modes and bumps. They used a Fourier Series representation with thousands of terms to fit the probability density function in $\Re^1$ and identify the modes, which is rather impressive given the computing resources at the time. Here we want to show the general mathematical insight of mode finding and modal clustering. In order to facilitate the discussion, we use the kernel density estimator as an exemplary density estimation algorithm. The basic spherical kernel estimator may be written compactly as:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\boldsymbol{x}-\boldsymbol{x}_i)\,, \tag{17}$$

where $\boldsymbol{x}_i \in \Re^d$ $(i = 1, 2, ..., n)$ are data points and $h$ is the smoothing parameter, which is applied to each dimension. We usually choose the kernel $K$ to be a standard Gaussian density, MN$(\mathbf{0}, \boldsymbol{I}_d)$ While a more general covariance matrix may be selected, it is equivalent to linearly transforming the data in a certain manner, so that this kernel is fully general.

An important result discovered by Silverman (1981) [100] was that with the Gaussian kernel, the number of modes decreases monotonically as the smoothing parameter $h$ increases, at least in one dimension; however, the result does not extend to higher dimensions; see Figure 6. The necessity (Silverman's paper showed sufficiency) to use Gaussian kernel for mode clustering in 1-D was proven by Babaud et al. (1994) [4]. Minnotte and Scott (1993) [83] created a tree-like graphical summary that captured all of the modes of a Gaussian kernel estimate as a function of the bandwidth $h$. It is called the "mode tree" in their original paper. Figure 5 displays the mode tree on the Old Faithful Geyser dataset and may be compared with the dendrogram hierarchical clustering tree on the same dataset. Since there is no probabilistic model for hierarchical clustering, it is difficult to analyze or make any probabilistic statement about features in the dendrogram. By contrast the mode tree is a powerful visualization tool for modes. Minnotte was able to test the veracity of individual modes by using the structure of the mode tree; see Minnotte and Scott (1993) [83] and Minnotte (1997) [81]. Since no single choice of bandwidth $h$ is likely to show all the potential modes at the same time, assessing the modes or the number of clusters is extremely challenging. In that sense the bootstrap is an appealing means to tackle this problem, due to its generality and lack of specific assumptions. Likewise since there is no probability model for the dendrogram clustering tree, it is difficult to assess dendrogram tree representation across bootstrap samples, since it is not obvious how to compare pairs of clustering trees due to the change in labels. The mode tree solves this problem. It is also worth noting that Silverman (1981) [100] suggests a conservative test of the null hypothesis in the univariate case that the unknown density has at most $k$ modes with a certain bandwidths $h$, which he called “critical bandwidths.” These are defined by an additional mode about to appear when the bandwidth is decreased further. This is where the horizontal dashed lines appear in the mode tree in Figure 5.

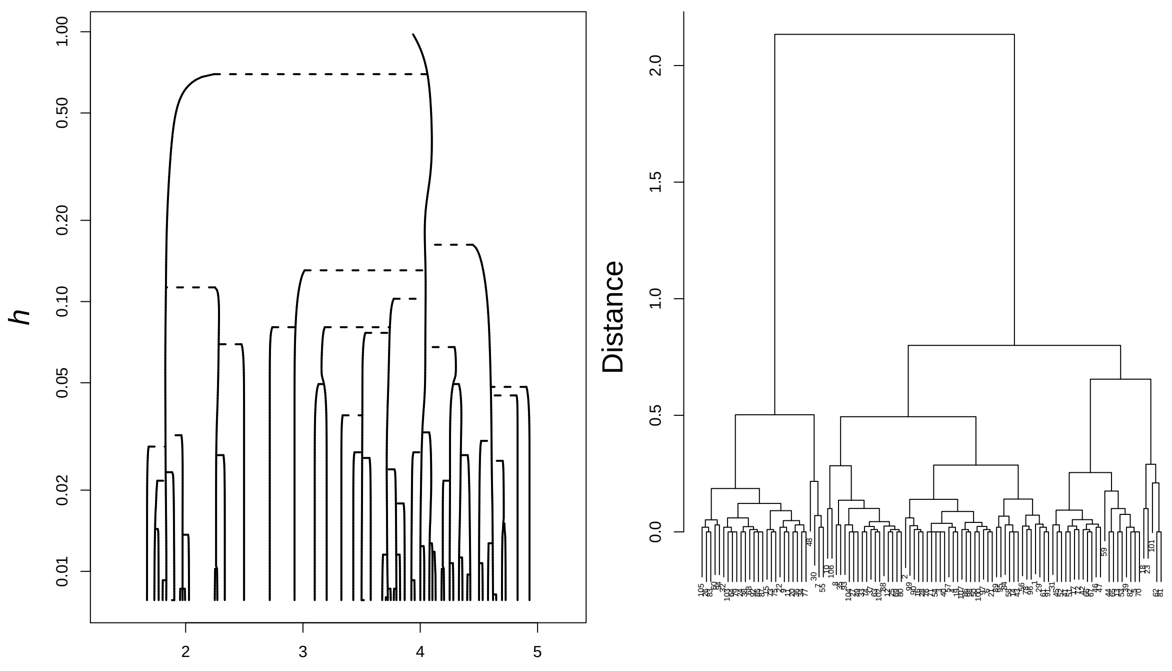


Figure 5: Illustration of the mode tree (left) and the dendrogram clustering tree (right) of the geyser eruption times dataset. Notice that the dendrogram (right) is created by hierarchical clustering based on the average linkage between modes.

In Silverman's study, he suggested counting the number of modes at those critical bandwidths across many bootstrap samples, assessing the veracity of that mode count by the distribution of the mode count in the resamples. It was noted by Matthews (1983) [77] that this approach might not work for complex densities. To be specific, if there is a true mode at a relatively low height that requires a small bandwidth to properly resolve, other taller modes may split into many noisy modes at the appropriate critical bandwidth, which make it very likely to mask the smaller mode across bootstrap samples (Scott, 2015 [97]). Also the final counting of modes will be influenced by any outliers because in the bootstrap sample the outliers show up with the probability of $1 - e^{-1} \approx 0.632$ of the time. To rephrase, a single "good" choice of the bandwidth $h$ is likely to both undersmooth and oversmooth in certain regions of the density, so that the count of modes is not very precise.

Minnotte (1997) [81] successfully showed that by testing the individual modes at critical bandwidths along the branch of the mode tree, one can appropriately evaluate the modes locally.

The idea is quite intuitive since nonparametric density estimation enjoys its success largely by being a local estimator. Other tree-like assessments including SiZer ([14], [15]) which is based on a scale-space representation. Erasto and Holmstrom (2005) [30] proposed a Bayesian version of SiZer, and Minnotte, et al. (1998) [82] proposed a "mode forest," which is an enhanced version of mode-tree to present a collection of bootstrapped mode trees. Another approach which was proposed for mode assessments is to look at contours around a mode and compute the "excess mass" in that region (Muller and Sawitzki (1991) [85] and Mammen et al. (1994) [74]). In high-dimensional problems where $d > 1$, the Hessian might be indefinite in some regions with a mixture of positive and negative eigenvalues. In that case the analysis becomes quite complicated and special cares have to be taken; see Marchette, et al. [75] and Wong (1993) [118].

In one dimension, there is a sufficient and necessary condition supporting that the number of modes is monotone for Gaussian kernel. The natural question is how the number of modes change in high-dimensions with the bandwidth $h$. Scott and Szewczyk (1997) [98] provided a counterexample that is shown in Figure 6; see Scott (2015) [97].

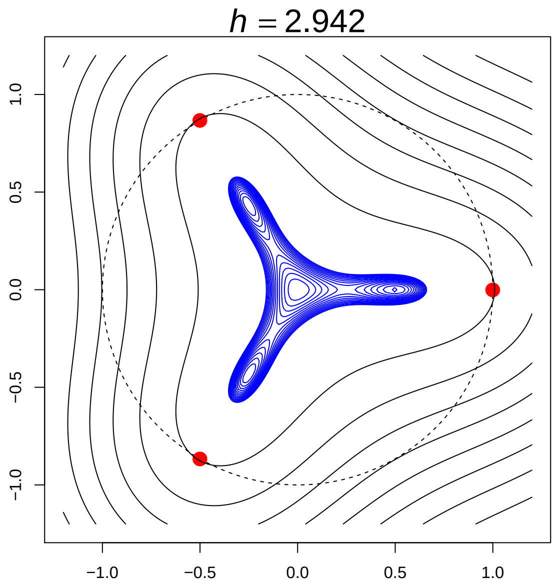


Figure 6: Example of 4 modes but only 3 data points (shown in red at vertices of an equilateral triangle) in two dimensions. The surface is very flat, which is highlighted by the blue contours around the origin. See the text for further details.

Figure 6 shows three data points at the vertices of an equilateral triangle, whose kernel density estimator generates three modes if $0 < h < 2.8248$ and one mode if $h > 2.9565$. However, in the very narrow range of bandwidths $2.8248 < h < 2.9565$, a fourth mode appears at the origin, one more than the true number of components. Figure 6 shows where the four modes are located and all are clearly visible. In their paper a circular covariance matrix is used, which corresponds to a mixture of three equally weighted Gaussian densities. Their result indicates that in high-dimensions, monotonicity also does not hold and that the range of bandwidths where this holds grows slightly with dimension. Other authors have found other counterexamples for points on a right angle rather than the regular mesh pattern here and unequal covariance matrices [11].

Extensive discussions of the multivariate versions of the mode tree have been done by Minnotte and Scott (1993) [83] and Klemela (2008, 2009) [60, 61]. Figure 7 illustrates the bivariate mode tree for the lagged geyser dataset. The trimodal feature is clearly visible. In more than 2 dimensions, placing $n = d + 1$ data points at the vertices of a regular polytope (the regular tetrahedron in $\Re^3$ for example) and using the $MN(\mathbf{0}, \boldsymbol{I}_d)$ kernel, we observe either 1, $d+1$, or $d+2$ modes. The range of bandwidths $h$ where the "phantom mode" at the origin is observed increases as $d$ grows by empirical observation. In our opinion, the possibility of phantom modes has little impact on clustering, but must be accounted for when programming and evaluating the mode tree. Assuming monotonicity can defeat a code when phantom modes appear.

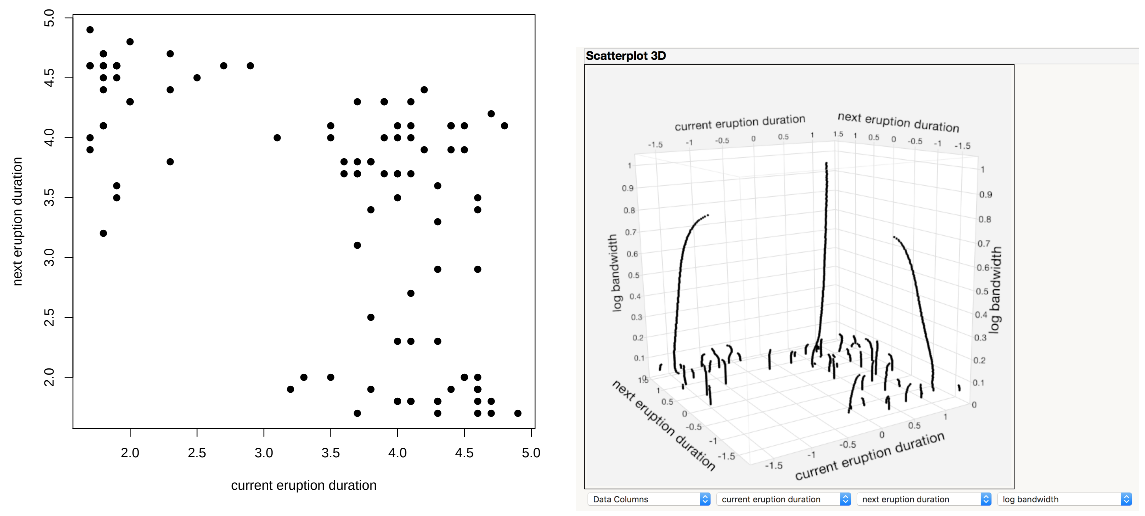


Figure 7: Bivariate scatterdiagram and bivariate mode tree for lagged geyser dataset. Kernel estimates for 201 choices of the logarithm of bandwidth $h$ (scaled to $(0, 1)$ were computed and the sample modes located. The data have three obvious clusters, which are visible in the scatterdiagram as well as the three long modal traces in the right frame.

There are several challenges associated with mode clustering, especially in high dimensions ($d \geq 4$). First of all, by the hard assignment of data points, it is difficult to evaluate the uncertainty of how well the data points are being clustered. How to visualize clusters in high dimensions ($d \geq 4$) also remains a difficult problem; see the ideas in Klemelä [60, 61] as well as Stuetzle and Nugent [103, 104]. As discussed before, the number of modes is heavily dictated by bandwidth $h$, and identifying the appropriate bandwidth $h$ for the kernel density estimator is not trivial. Using only one bandwidth everywhere with a kernel estimate is usually far from adequate as regions of oversmoothing and undersmoothing are inevitable. Thus, in high dimensions one cannot (entirely) avoid the likelihood that noisy modes will appear, even if local smoothing is attempted. How to assess those small noisy modes as well as missing or extra modes, and how to further denoise those modes are complicated questions, requiring sophisticated statistical analysis and further research. However, Chen et al. (2016) [17] has proposed a solution to all of those problems that leads to a complete and reliable approach for modal clustering. Notably they provided a soft assignment method that is able to capture the uncertainty of clustering, and define a measure of connectivity among clusters. They also proposed an estimate for that measure. Their method was proved to be consistent and is able to provide an enhanced capability for mode clustering in high-dimensions. Wasserman (2018) [115] discuss mode clustering in high dimensions in the context of topological data analysis, which represents a generalized collection of statistical methods that identify intrinsic structures of data by use of ideas of topology. Interested readers can also refer to Chen et al. (2016) [17]. We note, however, that they usually use a single bandwidth globally, so the results will be asymptotic to the globalized distributions unless the modes or features have nearly the same height and shape.

We conclude by observing that Ray and Lindsay [90] have given an elegant algorithm for finding all modes as $h$ varies by following the so-called "density ridges." These can also serve as a visualization tool, which are rather different than those of Klemela [60]. For $d > 2$, the Minnotte-Scott mode tree reverts to the dendrogram-like appearance as in Figure 5. The coalescence of adjacent (neighboring) modes may be determined by using the Ray-Lindsay algorithm. Interested readers can also refer to the work of Minnotte(2010) which uses high-order variants on kernel density estimation to test multimodality [84]. Finally, a partial survey of software available for modal clustering may be found in the Menardi survey [79].

## SUMMARY

Nonparametric density estimation is an active research field in machine learning and statistics. Conventional method such as Kernel Density Estimation (KDE) performs poorly for high-dimensional data ($d > 3$). For the real-world problems, ideally we want to have reliable density estimators for $3 < d \leq 50$. In this paper we reviewed some selected nonparametric density estimation algorithms which could potentially tackle high-dimensional problems. On the application side, modal clustering based on nonparametric density estimations enjoys high flexibility, adaptivity and performs well in a wide range of scenarios. The reviewed multivariate density estimation algorithms provide powerful building blocks for modal clustering. They are also able to compensate for some of the limitations of KDE. Future research should focus on developing more efficient, scalable, and reliable density estimation algorithms that work effectively in high dimensions. These algorithms ideally should lead to density functions that are as smooth as possible. They should also exhibit the property of effective local smoothing to minimize the likelihood of false or missing modes as correctly as possible.

## ACKNOWLEDGMENTS

The first author acknowledges the financial support from NSF and Rice University. The authors would also like to thank the two referees for helpful suggestions that refocused the article.

## References

[1] AITKIN, M. & WILSON, G.T. (1980). Mixture Models, Outliers, and the EM Algorithm, *Technometrics* **22**, 325–331.

[2] ARIAS-CASTRO, E. & LERMAN, G. & ZHANG, T. (2013) Spectral Clustering Based on Local PCA, arXiv preprint.

[3] AZZALINI, A. & TORELLI, N. (2007) Clustering Via Nonparametric Density Estimation, *Statistics and Computing*, **17**, 71–80.

[4] BABAUD, J. & WITKIN, A.P. & BAUDIN, M. & DUDA, R.O. (1994). Uniqueness of the Gaussian Kernel for Scale-Space Filtering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**, 26–33.

[5] BANFIELD, J.D. & RAFTERY, A.E. (1993) Model-Based Gaussian and Non-Gaussian Clustering, *Biometrics*, **49**, 803–821.

[6] Bengio, S. & Bengio, Y. (2000) Taking on the Curse of Dimensionality in Joint Distributions Using Neural Networks *IEEE Transactions on Neural Networks*, **11**, 550–557.

[7] Dinh, L. & Sohl-Dickstein, J. & Bengio, S. (2017) Density Estimation Using Real NVP. *Fifth International Conference on Learning Representations*.

[8] Bishop, C. & Svensen, M. & Williams, C. K. I. (1998) The Generative Topographic Mapping, *Neural Computation*, **10**, 215–234.

[9] Breese, J. S. & Heckerman, D. & Kadie, C. (1998) Empirical Analysis of Predictive Algorithms for Collaborative Filtering, *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence (UAI'98)*, 43–52.

[10] Carmichael, J.W. & George, J.A. & Julius, R.S. (1968) Finding Natural Clusters, *Syst. Zool.*, **17**, 144–150.

[11] Carreira-Perpinan, M. & Williams, C. (2003). On the Number of Modes of a Gaussian Mixture. *Scale Space Methods in Computer Vision* pp. 625–640, Springer.

[12] Chacón, J.E. & Duong, T. (2013) Data-Driven Density Derivative Estimation, with Applications to Nonparametric Clustering and Bump Hunting, *Electronic Journal of Statistics*, **7**, 499–532.

[13] Chacón, J.E. (2016) Mixture Model Modal Clustering, *Advances in Data Analysis and Classification*, pp. 1–26.

[14] Chaudhuri, P. & Marron, J.S. (1999). SiZer for Exploration of Structures in Curves, *Journal of the American Statistical Association*, **94**, 807–823.

[15] Chaudhuri, P. & Marron, J.S. (2000). Scale Space View of Curve Estimation, *Annals of Statistics*, **28**, 408–428.

[16] Chazal, F. & Guibas, L.J. & Oudot, S. & Skraba, P. (2013). Persistence-Based Clustering in Riemannian Manifolds, *J. ACM* **60**, 41–??.

[17] Chen, Y-C & Genovese, C.R. & Wasserman, L. (2016). A Comprehensive Approach to Mode Clustering, *Electronic Journal of Statistics*, **10**, 210–241.

[18] CHENG, Y. (1995). Mean Shift, Mode Seeking, and Clustering *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**,790–799.

[19] CHI, E. C.,& ALLEN, G. I. ,& BARANIUK, R. G. (2017) Convex Biclustering, *Biometrics*, **73**, 10–19

[20] CHI, E.C., & LANGE, K. (2015) Splitting Methods for Convex Clustering, *Journal of Computational and Graphical Statistics*, **24**, 994–1013

[21] CILIBRASI, R. & VITANYI, PAUL M.B. (2005). Clustering by Compression, *IEEE Transactions on Information Theory*, **51**, 1523–1545

[22] CLEVELAND, W. S. (1993). *Visualizing Data*. Summit: Hobart Press.

[23] CLEVELAND, W. S. (1994). *The Elements of Graphing Data*. Summit: Hobart Press, revised ed.

[24] COMANICIU, D. & MEER, P. (2002) Mean Shift: A Robust Approach Toward Feature Space Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**, 603–619.

[25] COX, D. R. (1972). Regression Models and Life Tables (With Discussion). *J. R. Statist. Soc.* B **34**, 187–220.

[26] DEMPSTER, A. P. & LAIRD, N. M. & RUBIN, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Royal Statist. Soc., Series B*, **39**, 1–38.

[27] DHILLON, I.S. & GUAN, Y. & KULIS, B. (2004) Kernel $k$-Means: Spectral Clustering and Normalized Cuts, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 551–556.

[28] DOBRA, A. & LENKOSKI, A. (2011). Copula Gaussian Graphical Models and Their Application to Modeling Functional Disability Data *The Annals of Applied Statistics* **5**, 969–993.

[29] EILERS, P. HC & MARX, B.D. (1996) Flexible Smoothing with B-Splines and Penalties, *Statistical Sciences*, textbf11, 89–102.

[30] ERÄSTÖ, P. & HOLMSTRÖM, L. (2005). Bayesian Multiscale Smoothing for Making Inferences About Features in Scatterplots, *Journal of Computational and Graphical Statistics*, **14**, 569–589.

[31] FAN, J. & PENG, H. (2004). Nonconcave Penalized Likelihood with a Diverging Number of Parameters. *Ann. Statist.* **32**, 928–61.

[32] FRALEY, C. & RAFTERY, A.E. (1999). MCLUST: Software for Model-Based Cluster Analysis, *Journal of Classification*, **16**, 297–306.

[33] FRALEY, C. & RAFTERY, A.E. (2002). Model-Based Clustering, Discriminant Analysis and Density Estimation, *J. Am. Stat. Assoc.*, **97**, 611–631.

[34] FREY, BRENDAN J. & HINTON, GEOFFREY E. & DAYAN, PETER (1996). Does the Wake-Sleep Algorithm Learn Good Density Estimators? *Advances in Neural Information Processing Systems 8* 661–670.

[35] FRIEDMAN, J.H. (1987). Exploratory Projection Pursuit. *Journal of the American Statistical Association*, **82**, 249–266.

[36] FRIEDMAN, J.H. & STUETZLE, W. (1981). Projection Pursuit Regression, *J. Amer. Statist. Assoc.* **76**, 817–823.

[37] FRIEDMAN, J.H. & STUETZLE, W. & SCHROEDER, A. (1984). Projection Pursuit Density Estimation, *J. Amer. Statist. Assoc.* **79**, 599–608.

[38] FRIEDMAN, J.H. & TUKEY, J.W. (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis, *IEEE Trans. in Computers* **C-23**, 881–890.

[39] FUKUNAGA, F. & HOSTETLER, L, (1975). The Estimation of the Gradient of a Density Function, With Applications in Pattern Recognition, *IEEE Transactions on Information Theory* **21**, 31–40.

[40] GERMAIN, M., & GREGOR, K,& MURRAY, I. & LAROCHELLE, H. (2015) MADE: Masked Autoencoder for Distribution Estimation., *Proceedings of the 32nd International Conference on Machine Learning* pages 881–889.

[41] GOOD, I.J. & GASKINS, R.A. (1980). Density Estimation and Bump-Hunting by the Penalized Likelihood Method Exemplified by the Scattering and Meteorite Data (With Discussion), *J. Amer. Statist. Assoc.* **75**, 42–73.

[42] GOODFELLOW, IAN & BENGIO, YOSHUA & COURVILLE, AARON (2016), Deep Learning, *MIT Press*, 2016

[43] GORDON, N., & SALMOND, D.J.,& SMITH, A.F.M. (1993) Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation, *IEE Proceedings on Radar and Signal Processing.* **140**, 107–113.

[44] GREGOR, KAROL & LECUN, YANN (2011). Learning Representations by Maximizing Compression, *Technical report, arXiv* pp. 1108–1169.

[45] HARTIGAN, J.A. (1975). *Clustering Algorithms.*, New York: J. Wiley & Sons.

[46] HASTIE, T. & TIBSHIRANI, R. & FRIEDMAN, J.H. (2001) *The Elements of Statistical Learning*, New York: Springer.

[47] HATHAWAY, R. J. (1985). A Constrained Formulation of Maximum-Likelihood Estimation for Normal Mixture Distributions, *Ann. Statist.* **13**, 795–800.

[48] HEARD, N. A., HOLMES, C. C. & STEPHENS, D. A. (2006). A Quantitative Study of Gene Regulation Involved in the Immune Response of Anopheline Mosquitoes: An Application of Bayesian Hierarchical Clustering of Curves. *J. Am. Statist. Assoc.* **101**, 18–29.

[49] HEBB, D.O. (1949). *The Organization of Behavior.* New York: Wiley & Sons.

[50] HJORT, N.L. (1986) On Frequency Polygons and Averaged Shifted Histograms in Higher Dimensions, Tech Report 22, Stanford University.

[51] HOLMSTROM, L. & HAMALAINEN, A. (1993) The Self-Organizing Reduced Kernel Density Estimator, *IEEE International Conference on Neural Networks*,

[52] IWATA, T. & YAMADA, M. (2016) Multi-view Anomaly Detection via Robust Probabilistic Latent Variable Models, *30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.*

[53] IZENMAN, A.J. (2008) *Modern Multivariate Statistical Techniques Regression, Classification, and Manifold Learning.* New York: Springer.

[54] JEE, J.R. (1987) Exploratory Projection Pursuit Using Nonparametric Density Estimation *Proceedings of the Statistical Computing Section of the American Statistical Association*, 335–339.

[55] JOHNSON, S. C. (1967) Hierarchical Clustering Schemes *Psychometrika*, **32**, 241–254.

[56] JONES, M. C. & SIBSON, R. (1987) What is Projection Pursuit? *Journal of the Royal Statistical Society. Series A (General)*, **150**, 1–37.

[57] JORDAN, M. I. (2004). Graphical Models, *Statistical Science* **19**, 140–155.

[58] KINGMA, D.P. & SALIMANS, T. & JOZEFOWICZ, R. & CHEN, X. & SUTSKEVER, I. & WELLING, M. (2016) Improved variational inference with Inverse Autoregressive Flow., *Advances in Neural Information Processing Systems 29*, pages 4743–4751.

[59] KONG, A., & LIU, J. S., & WONG, W. H. (1994) Sequential Imputations and Bayesian Missing Data Problems, *Journal of the American Statistical Association*, **89**, 278–288.

[60] KLEMELÄ, J. (2008). Mode Trees for Multivariate Data. *J. Comput. Graph. Statist.*, **17**, 860–869.

[61] KLEMELÄ, J. (2009). *Smoothing of Multivariate Data: Density Estimation and Visualization.* Hoboken, NJ: John Wiley & Sons.

[62] KOLLER, D. & FRIEDMAN, N. (2009). Probabilistic Graphical Models: Principles and Techniques, *MIT Press.*

[63] LAROCHELLE, H. & MURRAY, I. (2011). The Neural Autoregressive Distribution Estimator, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. pp. 29–37.

[64] LI, DANGNA. & YANG, KUN. & AND WONG, WING H. (2016). Density Estimation via Discrepancy Based Adaptive Sequential Partition, *30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.*

[65] LI, JIA. & RAY, SURAJIT & LINDSAY, BRUCE G. (2007). A Nonparametric Statistical Approach to Clustering via Mode Identification, *Journal of Machine Learning Research* **8**, 1687–1723.

[66] LIU, ALEXANDER Y. & LAM, DUNG N. (2012). Using Consensus Clustering for Multi-view Anomaly Detection, *IEEE CS Security and Privacy Workshops*, 117–124.

[67] LIU, H. & HAN, FANG & YUAN, MING& LAFFERTY, JOHN & WASSERMAN, LARRY (2012). High-Dimensional Semiparametric Gaussian Copula Graphical Models, *The Annals of Statistics* **40**, 2293–2326.

[68] LIU, J. S. (2001). Monte Carlo Strategies in Scientific Computing, *Springer Series in Statistics, New York: Springer*.

[69] LOPEZ, TOMAS SANCHEZ. & BRINTRUP, ALEXANDRA. & ISENBERG, MARC-ANDRE. & MANSFELD, JEANETTE. (2011). Resource Management in the Internet of Things: Clustering, Synchronisation and Software Agents, *Architecting the Internet of Things*, pp. 159–193.

[70] LU, LUO. & JIANG, HUI. & AND WONG, WING H. (2013). Multivariate Density Estimation by Bayesian Sequential Partitioning, *Journal of the American Statistical Association* **108**, 1402–1410.

[71] MACQUEEN, J.B. (1967). Some Methods for Classification and Analysis of Multivariate Observations, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability 1* University of California Press, pp. 281–297.

[72] MCLACHLAN, G. & KRISHNAN, T. (2008). *The EM Algorithm and Extensions, 2nd Edition* Hoboken, NJ: John Wiley & Sons.

[73] MAGDON-ISMAIL, M. & ATIYA, A. (1998). Neural Networks for Density Estimation, *In NIPS* pp. 522–528.

[74] MAMMEN, E. & MARRON, J.S. & FISHER, N.I. (1994). Asymptotics for Multimodality Tests Based on Kernel Density Estimates, *Prob. Theory Related Fields* **91**, 115–132.

[75] MARCHETTE, D.J. & PRIEBE, C.E., & ROGERS, G.W. & AND WEGMAN, E.J. (1996). Filtered Kernel Density Estimation, *Comp. Statist.* **11**, 112.

[76] MARCHETTI, Y. & NGUYEN, H. & BRAVERMAN, A. & CRESSIE, N. (2018). Spatial Data Compression via Adaptive Dispersion Clustering, *Computational Statistics and Data Analysis*, **117**, 138–153.

[77] MATTHEWS, M.V. (1983). On Silverman's Test for the Number of Modes in a Univariate Density Function, Honors Bachelor's Thesis, Harvard University.

[78] MEILA, M. & SHI, J. (2000). Learning Segmentation by Random Walks, *Neural Information Processing Systems*, **13**, 873–879.

[79] MENARDI, G. (2016). A Review on Modal Clustering, *International Statistical Review* **84**, 413–433.

[80] MINKA, THOMAS. (2005). Divergence Measures and Message Passing, *MSR- Technical Report- 2005-173*.

[81] MINNOTTE (1997). Nonparametric Testing of the Existence of Modes. *Ann. Statist.* **25**, 1646–1660.

[82] MINNOTTE, M.C. & MARCHETTE, D.J. & WEGMAN, E.J. (1998). The Bumpy Road to the Mode Forest *J. Comp. Graph. Statist.* **7** 239–251.

[83] MINNOTTE, M.C. & SCOTT, D.W. (1993). The Mode Tree: A Tool for Visualization of Nonparametric Density Features, *J. Comp. Graph. Statist.* **2** 51–68.

[84] MINNOTTE, M.C. (2010). Mode Testing via Higher-Order Density Estimation, *Computational Statistics* **25** 391-407.

[85] MÜLLER, D.W. & SAWITZKI, G. (1991). Excess Mass Estimates and Tests for Multimodality, *J. Amer. Statist. Assoc.* **86**, 738–746.

[86] PAPAMAKARIOS, G. & PAVLAKOU, T. & MURRAY, I. (2017). Masked Autoregressive Flow for Density Estimation, *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA* pages 2335–2344.

[87] PEARSON, K. (1894). Contributions to the Mathematical Theory of Evolution, *Philosophical Trans. Royal Society London (A)* **185**, 71–110.

[88] PHAM, MANH CUONG. & CAO, YIWEI. & KLAMMA, RALF. & JARKE, MATTHIAS. (2011). A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis, *Journal of Universal Computer Science*, **17**, 583–604.

[89] R DEVELOPMENT CORE TEAM (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0, http://www.R-project.org.

[90] RAY, SURAIJIT& LINDSAY, BRUCE G. (2005) The Topograph of Multivariate Normal Mixtures Annals of Statistics **13**, 2042–2065.

[91] REZENDE, D. J. & MOHAMED, S. (2015). Variational Inference with Normalizing Flows, *Proceedings of the 32nd International Conference on Machine Learning* pages 1530–1538.

[92] ROEDER, K. (1990). Density Estimation with Confidence Sets Exemplified by Superclusters and Voids in the Galaxies. *J. Amer. Statist. Assoc.* **85**, 617–624.

[93] SALIMANS, T. & KARPATHY, A. & CHEN, X. & KINGMA, D.P. (2017) PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, *Fifth International Conference on Learning Representations (ICLR)*.

[94] SCOTT, D. W. & TAPIA, R. A. & THOMPSON, J. R. (1977). Kernel Density Estimation Revisited, *J. Nonlinear Analysis Theory Meth. Applic.* **1**, 339–372.

[95] SCOTT, D. W. (1979). On Optimal and Data-Based Histograms, *Biometrika* **66**, 605–610.

[96] SCOTT, D. W. (1985). Frequency Polygons: Theory and Application, *J. Amer.'Statist. Assoc.* **80**, 348–354.

[97] SCOTT, D. W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualization*. Hoboken: John Wiley & Sons, 2nd edition.

[98] SCOTT, D. W. & SZEWCZYK, W. F. (1997). Bumps Along the Road Towards Multivariate Mode Trees, NSF Workshop on Bumps, Jumps, Clustering and Discrimination May 11-14, 1997, Houston, TX.

[99] SHI, J. & MALIK, J. (2000). Normalized Cuts and Image Segmentation, *IEEE Transactions on PAMI*, **22**, Aug 2000.

[100] SILVERMAN, B. W. (1981). Using Kernel Density Estimates to Investigate Multimodality, *J. Roy. Statist. Soc. B* **43**, 97–99.

[101] SILVERMAN, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

[102] STONE, C. J. (1994). The Use of Polynomial Splines and Their Tensor Products in Multivariate Function Estimation, *The Annals of Statistics*, **22**, 118–171.

[103] STUETZLE, W. (2003). Estimating the Cluster Tree of a Density by Analyzing the Minimal Spanning Tree of a Sample. *J. Classification* **20**, 25–47.

[104] STUETZLE, W. & NUGENT, R. (2010). A Generalized Single Linkage Method for Estimating the Cluster of a Density, *J. Computational & Graphical Statistics* **19**, 397–418.

[105] SU, XIAOYUAN. & KHOSHGOFTAAR, TAGHI M. (2009). A Survey of Collaborative Filtering Techniques, *Advances in Artificial Intelligence*, 421425–421443.

[106] SVENSEN, M. (1998). GTM: The Generative Topographic Mapping. *Ph.D. thesis, Aston University, Birmingham, UK.*

[107] TASOULIS, S. K., ET AL. (2012). Density Based Projection Pursuit Clustering, *Evolutionary Computation (CEC), 2012 IEEE Congress on. IEEE* pp. 1–12.

[108] TUFTE, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire: Graphics Press.

[109] URIA, BENIGNO & MURRAY, IAIN & LAROCHELLE, HUGO (2013). RNADE: The Real-Valued Neural Autoregressive Density Estimator *Advances in Neural Information Processing Systems 26*.

[110] URIA, BENIGNO& MURRAY, IAIN& LAROCHELLE, HUGO (2014). A Deep and Tractable Density Estimator *Proceedings of the 31st International Conference on Machine Learning, JMLR W&CP* **32**, 467–475.

[111] URIA, BENIGNO& COTE, MARC-ALEXANDRE& GREGOR, KAROL& MURRAY, IAIN& LAROCHELLE, HUGO (2016). Neural Autoregressive Distribution Estimation *Journal of Machine Learning Research* **17**, 1–37.

[112] WAINWRIGHT, M. (2008) Graphical Models, Exponential Families, and Variational Inference, *Foundations and Trends in Machine Learning*, **1**, 1–305.

[113] WAND, M. P. (2017) Fast Approximate Inference for Arbitrarily Large Semiparametric Regression Models via Message Passing, *Journal of the American Statistical Association*, **112**, 137–168.

[114] WANG, X. & WANG, Y. (2015) Nonparametric multivariate density Estimation Using Mixtures, *Statistics and Computing*, **25**, 349–364.

[115] WASSERMAN, L. (2018). Topological Data Analysis, *Annual Review of Statistics and Its Application* **5**: 501–532.

[116] WONG, WING H. (2014). Multivariate Density Estimation and its Applications, *Conference in honor of the 80th birthday of Professor Grace Wahba, June 2014, Madison, Wisconsin.*

[117] WONG, WING H. & AND MA, LI. (2010). Optional Polya tree and Bayesian inference, *Annals of Statistics* **38**, 1433–1459.

[118] WONG, Y. (1993). Clustering Data by Melting. *Neural Computation* **5**, 89–104.

[119] YEDIDIA, J. S. & FREEMAN, W. T. & WEISS, Y. (2003). Understanding Belief Propagation and its Generalizations, *Exploring artificial intelligence in the new millennium 8*, pp. 236–239.

[120] ZHU, JUN-YAN. & PARK, TAESUNG. & ISOLA, PHILLIP. & EFROS, ALEXEI A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, *IEEE International Conference on Computer Vision (ICCV)*.