Anomaly detection with kernel density estimation on manifolds

Keywords: manifold learning, variable bandwidth, Riemannian metric, highest density region, Gaussian kernels

1 Introduction

In manifold learning, the underlying idea is that the data lies on a low-dimensional smooth manifold which is embedded in a high-dimensional space. One of the fundamental objectives of manifold learning is to explore the geometry of the dataset, including the distances between points and volumes of regions of data. These intrinsic geometric attributes of the data, such as distances, angles, and areas, however, can be distorted in the low-dimensional embedding, leading to failure to recover the geometry of the manifold (Goldberg et al. 2008). To tackle this problem and measure the distortion incurred in manifold learning, Perrault-Joncas & Meila (2013) propose the Metric Learning algorithm to augment any existing embedding output with geometric information in the Riemannian metric of the manifold itself. By applying the Metric Learning algorithm, the outputs of different manifold learning methods can be unified and compared under the same framework, which would highly benefit in improving the effectiveness of the embedding.

The Riemannian metric defined at each point of the manifold is used to compute the geometric quantities, including angle, length, and volume, of the low-dimensional manifold embedding in any coordinate system, and be further applied to correct the distortion caused by the manifold learning algorithms. In variable kernel density estimate, the bandwidth matrix H is also defined to control the amount of smoothing for each data point. Therefore, if we could replace the bandwidth matrix with the Riemannian metric, we could further get the kernel density estimation of the manifold $\mathcal{M}$. This kernel density estimate can then be used to produce the highest density region plots (Hyndman 1996) for anomaly detection.

By applying an existing manifold learning algorithm to the data $X \in \mathbb{R}^r$ with n observations, a low-dimensional embedding $f_n \in \mathbb{R}^d$ can be computed. Most manifold learning methods involve the construction of the nearest neighbor graph based on which the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ is built. The Laplacian is quite useful because it can be coordinate-free while containing all the important geometry. Perrault-Joncas & Meila (2013) have stated one way to compute the approximated $\Delta_{\mathcal{M}}$

with a discrete consistent estimator, the geometric graph Laplacian $\mathcal{L}_{\varepsilon,n}$ (Zhou & Belkin 2011), where ε is the radius parameter for the nearest neighbor graph. The graph Laplacian together with the embedding can be used in the Metric Learning algorithm to achieve the augmented embedding with the Riemannian metric (f_n, g_n). The highlighted two steps in Figure 1 are the main contributions of this main chapter, replacing the bandwidth matrix H_i with the Riemannian metric g_i for each point in variable kernel density estimate, and computing the highest density region plots based on the density estimates, $\hat{f}()$, for anomaly detection.

The rest of the paper is organized as follows. In Section 2, we present the proposed algorithm to detect anomalies based on variable kernel density estimates of manifold embeddings. In this section, we provide justification for the use of Riemannien matrix as the bandwidth of variable kernel density estimation, including the comparison with fixed bandwidth. Section 3 is composed with two simulations with the proposed algorithm; the first deals with a 2-dimensional meta data embedded in 3-D and the second with a 100-dimensional meta data. Section 4 contains the application to visualize and identify anomalies in the [TODO] dataset. Conclusions and discussions are presented in Section 5.

2 Variable kernel density estimation with manifold embeddings

In this section, we introduce the proposed method to detect anomalies based on the kernel density estimates of manifold learning embeddings where the Riemannian matrix is used as the pointwise variable bandwidth to measure the direction and angle of the distortion of the low-dimensional embeddings. For a high-dimensional data set, various manifold learning algorithms including ISOMAP, LLE, Laplacian Eigenmaps, t-SNE, and UMAP, are applied to get a low-dimensional embedding. The manifold learning algorithms map the points through nonlinear functions that stretches some regions of the space while shrinks others. Perrault-Joncas & Meila (2013) gives us an idea of how to measure the direction and angle of the distortion using the Riemannian metric and the Riemannian metric is a positive simi-definite square matrix for each data point. To learn the distribution of the low-dimensional embedding, we use the kernel density estimation with the bandwidth matrix being the Riemannian metric. The outliers could then be defined as the points with lowest density estimates. The proposed schematic is shown in Figure 1.

To start with, we introduce the notations in this manuscript. Then we introduce the multivariate kernel density estimation method with variable bandwidth matrix and the metric learning algorithm to derive the pointwise Riemannian metrix. Finally, we propose our novel method to detect anomalies for high-dimensional data set.

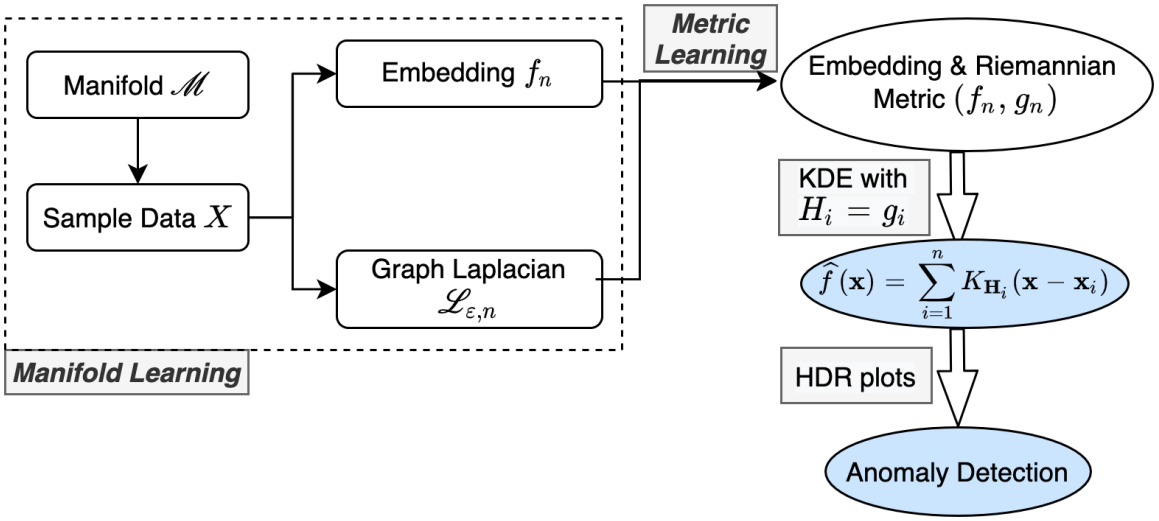

Figure 1: *The proposed schematic for variable kernel density estimation with recovered geometry.*

2.1 Notations

Manifold learning finds a d-dimensional representation of data that lie on a manifold $\mathcal{M}$ embedded in a p-dimensional ambient space with $d \ll p$. We will denote the original data (or 'input' points) as x_i, $i = 1, \ldots, N$, where $x_i \in \mathbb{R}^p$, while the low-dimensional representation (or 'output' points) will be denoted as y_i, $i = 1, \ldots, N$, where $y_i \in \mathbb{R}^d$. Where two subscripts are used (e.g. x_{ih} or y_{ih}), the second subscript refers to the h^{th} coordinate or dimension of the data.

2.2 Two dimensional kernel density estimation

For a bivariate random sample $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n$ drawn from a density f, the kernel density estimate is defined by

$$\hat{f}(\boldsymbol{x}; \boldsymbol{H}) = n^{-1} \sum_{i=1}^{n} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{X}_i)$$

where $\boldsymbol{x} = (x_1, x_2)^T$ and $\boldsymbol{X}_i = (X_{i1}, X_{i2})^T, i = 1, 2, \ldots, n$. Here $K(\boldsymbol{x})$ is the kernel which is a symmetric probability density function, $\boldsymbol{H}$ is the bandwidth matrix which is symmetric and positive-definite, and $K_{\boldsymbol{H}}(\boldsymbol{x}) = |\boldsymbol{H}|^{-1/2} K(\boldsymbol{H}^{-1/2}\boldsymbol{x})$. The choice of K is not crucial: we take $K(\boldsymbol{x}) = (2\pi)^{-1} \exp(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x})$ the standard normal throughout.

In contrast, the choice of $\boldsymbol{H}$ is crucial in determining the performance of $\hat{f}$. The most common parameterizations of the bandwidth matrix are the diagonal and the general or unconstrained which has no restrictions on $\boldsymbol{H}$ provided that $\boldsymbol{H}$ remains positive definite and symmetric, that is

$$\boldsymbol{H} = \begin{bmatrix} h_1^2 & 0 \\ 0 & h_2^2 \end{bmatrix} \text{ or } \boldsymbol{H} = \begin{bmatrix} h_1^2 & h_{12} \\ h_{12} & h_2^2 \end{bmatrix}.$$

This latter parameterization allows kernels to have an arbitrary orientation whereas the former only allows kernels which are oriented to the co-ordinate axes.

An alternative way to think of kernel estimation is that kernel densities 'borrow strength' from nearby points and the bandwidth determines what is "nearby". If the bandwidth is large then all points are "nearby" and we get an overly smooth kernel density estimate. The interesting thing about a bandwidth matrix is that it allows for different notions of what is "nearby" along different coordinates and even along diagonal directions.

2.3 Use riemannian matrix as variable bandwidth

The Riemannian estimated using the method of Perrault Joncas and Meila (2009) gives some idea of the distortion of an embedding (or so they claim). Mapping the points through a non-linear function "stretches" some regions of space and "shrinks" others. The Riemannian gives us an idea of the direction and angle of this stretching. The Riemannian is quite a technical concept but an important thing to understand is that the estimate that comes out of Perrault Joncas algorithm is a square matrix.

We saw how points that are far apart in the embedding may not have been so far apart on the original manifold. The Riemannian gives us some way of correcting this. Similarly a bandwidth matrix in a kernel density estimate is all about determining the "directions" in which there should be more or less "closeness". So the basic idea is to replace the kernel density estimate with

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{N} K_{\boldsymbol{H}_i}(\boldsymbol{x} - \boldsymbol{x}_i) K_{\boldsymbol{H}_i}(\boldsymbol{x} - \boldsymbol{x}_i) = (2\pi)^{-d/2} |\boldsymbol{H}_i|^{-1/2} \exp\left[-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_i)' \boldsymbol{H}_i^{-1} (\boldsymbol{x} - \boldsymbol{x}_i) \right]$$

where H_i is either the Riemannian or the inverse of the Riemannian (I am not totally sure which one). Notice that the bandwidth matrix is different for each point. This makes it a kernel density estimate with local smoothing, which is quite interesting, but we should take care to understand the properties of such things.

The Riemannian metric g is a symmetric and positive definite tensor field which defines an inner product $<,>_g$ on the tangent space $T_p\mathcal{M}$ for every point $p \in \mathcal{M}$. If the inner product of the tangent space is known for a given geometry, the Riemannian metric is a good measure to recover the geometry of manifold. The Metric Learning algorithm (Perrault-Joncas & Meila 2013) then augment the embedded manifold with the Riemannian metric and produce a Riemannian manifold $(\mathcal{M}, g)$.

To recover the original geometry of the manifold, we need to know what the inner product corresponds to in the embedding. The inner product between two vectors $u, v \in T_p\mathcal{M}$,

$< u, v >_g = g_{ij} u^i v^j$[1], can be used to define some geometric quantities, such as the vector norm $\|u\| = \sqrt{< u, v >_g}$ and the angle between two vectors $\cos\theta = \frac{<u,v>_g}{\|u\|\|v\|}$ in the tangent space. Therefore, for each point $p \in \mathcal{M}$ in any coordinate system, the Riemannian metric g is a $d \times d$ symmetric positive definite matrix, where d is the dimension of the manifold.

The line element and volume element of the full manifold or a subset of the manifold can also be computed from g. The arc length of a curve $c \in \mathcal{M}$ is defined as

$$l(c) = \int_a^b \sqrt{g_{ij} \frac{dx^i}{dt} \frac{dx^j}{dt}} dt,$$

where $(x^1, \ldots, x^d)$ are the coordinates of chart (U, x) and $c(t)$ is a function mapping $[a, b]$ to $\mathcal{M}$. While the volume of $V \subset \mathcal{M}$ is computed by

$$Vol(V) = \int_V \sqrt{\|g\|} dx^1 \ldots dx^d.$$

Both the concepts of distance and volume are relevant to kernel density estimation.

The aim of the project is not to evaluate different manifold learning algorithms. Although ISOMAP worked OK here it is still distorted and in real examples you will never 'know' that the data are uniform on the sphere. Instead the idea of this project is to do kernel density estimation in a way that takes distortion into account.

2.3.1 Metric Learning algorithm

Perrault-Joncas & Meila (2013) propose the Metric Learning algorithm which mainly involves four main steps.

As pointed out by Perrault-Joncas & Meila (2013), if the embedding dimension s is larger than the manifold intrinsic dimension d, the rank of the embedding metric $h_n(p)$ is d; otherwise, the Riemannian metric g_n will be returned. This algorithm is also implemented in a Python library *megaman* (McQueen et al. 2016). It is designed to apply the manifold learning methods to large-scale data sets, as well as computing the Riemannian metric of the manifold.

Learn metric algorithm

2.4 Proposed algorithm

Now we present our proposed method for anomaly detection based on variable kernel density estimates.

[1] Here the Einstein notation is used where superscripts denote summation over i and j

Algorithm 1: Learn metric algorithm

Input : high-dimensional data $x_i \in \boldsymbol{R}^p$ for all $i = 1, \ldots, N$
Output : low-dimensional data $y_i \in \boldsymbol{R}^d$ and its Riemannian metric h_i for all $i = 1, \ldots, N$
parameter : embedding dimension d, bandwidth parameter $\sqrt{\varepsilon}$, manifold learning algorithm
optimization parameter: manifold learning parameters

1: Construct a weighted neighborhood graph $\mathcal{G}_{w,\varepsilon}$ with weight matrix W where $w_{i,j} = \exp(-\frac{1}{\varepsilon}\|x_i - x_j\|^2)$ for data points $x_i, x_j \in \mathbb{R}^p$;
2: Calculate the $N \times N$ geometric graph Laplacian $\widetilde{\mathcal{L}}_{\varepsilon,N}$ by

$$\widetilde{\mathcal{L}}_{\varepsilon,N} = 1/(c\varepsilon)(\widetilde{D}^{-1}\widetilde{W} - I_N),$$

where $\widetilde{D} = diag\widetilde{W}\mathbf{1}$, $\widetilde{W} = D^{-1}WD^{-1}$, and $D = diagW\mathbf{1}$;
3: Embed each data point $x \in \mathbb{R}^p$ to embedding coordinates $y(x) = (y^1(x), \ldots, y^d(x))$ by any existing manifold learning algorithm;
4: Obtain the matrix $\tilde{h}(x)$ at each point by applying the graph Laplacian $\widetilde{\mathcal{L}}_{\sqrt{\varepsilon},N}$ to the embedding coordinates y with each element being

$$\tilde{h}^{ij} = \frac{1}{2}\left[\tilde{\mathcal{L}}_{\varepsilon,N}\left(y_i \cdot y_j\right) - y_i \cdot \left(\tilde{\mathcal{L}}_{\varepsilon,n} y_j\right) - y_j \cdot \left(\tilde{\mathcal{L}}_{\varepsilon,n} y_i\right)\right];$$

5: Calculate the Riemannian metric $h(x)$ as the rank d pseudo inverse of $\tilde{h}(x)$ with

$$h(x) = Udiag1/(\Lambda[1:d])U',$$

where $[U, \Lambda]$ is the eigendecomposition of matrix $\tilde{h}(x)$, and U is the matrix of column eigenvectors ordered by the eigenvalues Λ in descending order.

Algorithm 2: Variable kernel density estimates with Riemannian metric

Input : high-dimensional data x_i for all $i = 1, \ldots, N$
Output : outliers embedding coordinates $y_1, \ldots, y_{n\_outliers}$ with their estimated densities $f_1, \ldots, f_{n\_outliers}$
parameter: number of outliers *n_outliers*, embedding dimension d

1: For all $i = 1, \ldots, N$, compute the d-dimensional embeddings y_i with any exsiting manifold leanring algorithms and the corresponding Riemannian metric h_i using the Learn metric algorithm with inputs d and $\sqrt{\varepsilon} = 0.4$ and $c = 0.25$ for heat kernels;
2: Set the variable bandwidth for each point as $\boldsymbol{H}_i = h_i$;
3: Compute the kernel density estimates for each point as

$$\hat{f}(\boldsymbol{y}) = \sum_{i=1}^{N} (2\pi)^{-d/2} |\boldsymbol{H}_i|^{-1/2} \exp\left[-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}_i)' \boldsymbol{H}_i^{-1} (\boldsymbol{y} - \boldsymbol{y}_i)\right];$$

4: Reorder the embedding coordinates y according to the density estimates $f(y)$ and subset the top *n_outliers* as the outliers.

Now that we have proposed a way to take into account the distortion of manifold in kernel density estimate, it would be straightforward to produce the highest density region plots [HDR plots; Hyndman (1996)] which are also computed using kernel density estimate of the embedding.

3 Simulation

In this section, we exam two scenarios for both low and high dimensions to test our proposed algorithm. For visualization purpose, Section 3.1 presents a 2-D meta data example. We first simulate the data of size $N = 2,000$ from a mixture of four Gaussian kernels with the same covariance but different means, each consisting of 500 points. Different mapping functions are then applied to the 2-D meta data to be mapped in a 3-D feature space, which gives the higher-dimensional input for different manifold learning algorithms, including ISOMAP, LLE, Laplacian Eigenmaps, t-SNE, and UMAP. The embedded dimension is set as $d = 2$, same as the meta data dimension. This enables us to compare the manifold learning embedding with the true meta data. We could now apply Algorithm 2 to get the density estimates of all data points and further detect anomalies. As a high-dimensional example, the second simulation in Section 3.2 is based on a 5-D meta data of size $N = 10,000$ embedded in a 100-D space and the corresponding embedding dimension is $d = 5$.

3.1 2-D Meta data from a Gaussian Mixture Model

We first generate a 2-dimensional data of size $N = 2000$ from a Gaussian mixture model with four components with different means $\boldsymbol{\mu_1} = (0.25, 0.25)', \boldsymbol{\mu_2} = (0.25, 0.75)', \boldsymbol{\mu_3} = (0.75, 0.25)', \boldsymbol{\mu_4} = (0.75, 0.75)'$ and the same variance-covariance matrix $\boldsymbol{\Sigma}_i = diag(0.02, 0.02), i = 1, 2, 3, 4$. The mixture proportions are equally set as $\pi_i = 0.25, i = 1, 2, 3, 4$. Then the mixture Gaussian mixture density function is a weighted linear combination of the four component Gaussian densities as

$$P(\boldsymbol{X} = \boldsymbol{x}) = \sum_{i=1}^{4} \pi_i \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{-1/2}} \exp\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_i})'\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x} - \boldsymbol{\mu_i})\}. \tag{1}$$

Figure 2 shows the 2-dimensional meta data and the colors indicate the true density of all data points calculated from (1), with brighter colors showing higher densities and darker colors showing lower densities. We then define outliers as points with lowest densities shown in black and typical points with highest densities shown in yellow. Based on the true density plot, the outliers are scattered in the middle and the outer area of the whole structure, while typical points are near the means of four kernels. These are *true outliers* to be compared with outliers from the kernel density estimates.

3.1.1 Swiss roll mapping

Given the 2-D meta data, multiple mapping functions could be applied to embed the data in a 3-D space. One of the most famous example in manifold learning is the swiss roll data, with the mapping function in (2). The two-dimensional meta data $(\boldsymbol{X}_1, \boldsymbol{X}_2)'$ is transformed into the three-dimensional data $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})'$, shown in the left plot of Figure 3. The four colors in the mappings represents the

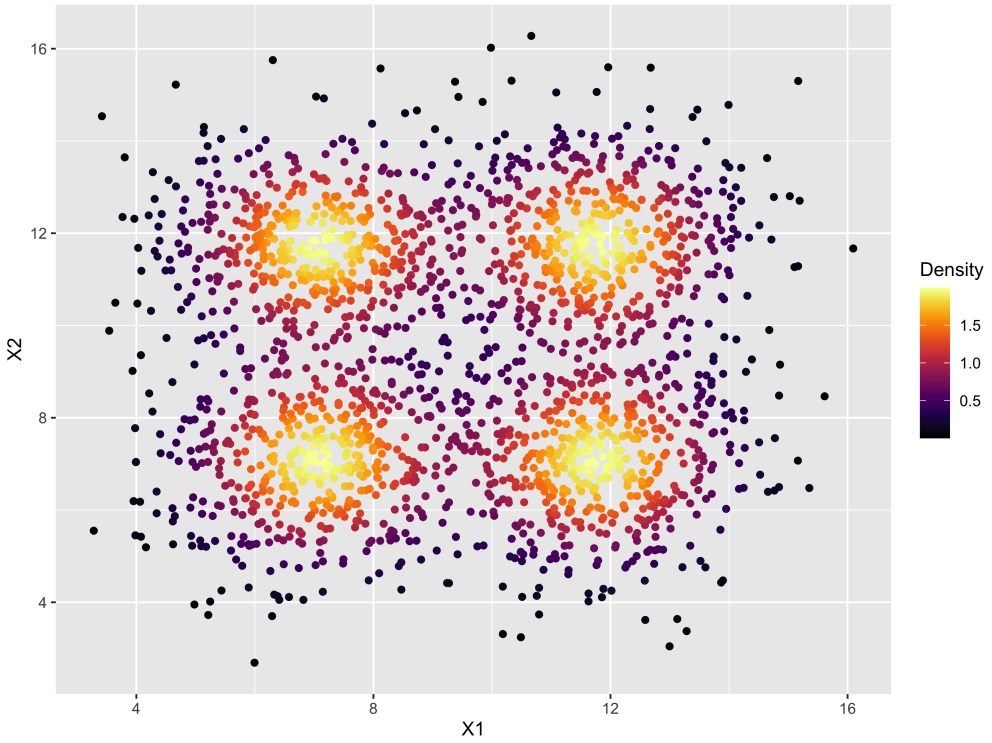

Figure 2: *True density of the Gaussian mixture model of four kenels with means* $(0.25, 0.25), (0.25, 0.75), (0.75, 0.25), (0.75, 0.75)$ *and the same variance-covariance matrix* $diag(0.02, 0.02)$.

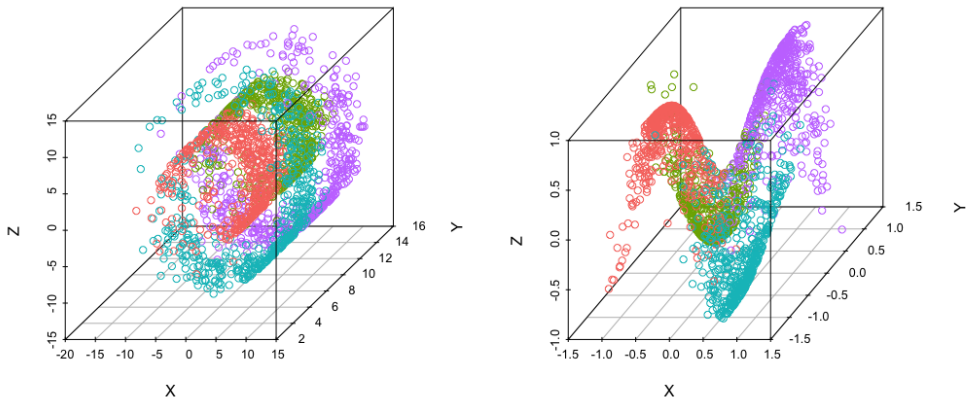

Figure 3: *3-D Mappings of the meta data with colors indicating four kernels. Left: swiss roll mapping. Right: twin peak mapping.*

Table 1: *Correlation between true density ranking and estimated density ranking for different manifold learning embeddings.*

	ISOMAP	LLE	Laplacian.Eigenmaps	t.SNE	UMAP
Variable bandwidth	0.0696	**0.400**	-0.2357	0.023	**0.0138**
Fixed bandwidth	**0.2798**	0.351	**0.0141**	**0.367**	-0.0110

four Gaussian kernels used to generate the meta data $(\boldsymbol{X}_1, \boldsymbol{X}_2)'$.

$$\begin{cases} X = X_1 \cos X_1, \\ Y = X_2, \\ Z = X_1 \sin X_1. \end{cases} \tag{2}$$

Now we are able to apply different manifold learning algorithms to $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})'$ and reduce the dimension back to $d = 2$, and further estimate the density of the 2-D embedding. According to the density estimates, we could rank the data points and then identify which observations lie in an highest density region of specified coverage, eg. 1%, 5%, 50%, 99%, >99%. For each of the five manifold learning methods, namely ISOMAP, LLE, Laplacian Eigenmaps, t-SNE, and UMAP, Figure 4 presents the 2-D embedding plot in the same row, with the colors indicating the densities levels, the left column for true densities from the Gaussian mixture model, the middle column for highest density region plots with densities from our proposed variable KDE method, and the right for similar HDR plots with densities from KDE with fixed bandwidth. The top twenty outliers with lowest densities are highlighted in black with point indexes in blue. From Figure 2 and the data generating process, we know that there are four highest density regions. However, in all manifold learning embeddings colored with true densities (left column in Figure 4), except for LLE, the number of highest density regions are not the same as the meta data. When comparing the number of HDRs for variable and fixed bandwidth (middle and right column in Figure 4), our proposed method with variable bandwidth outperforms fixed bandwith for ISOMAP, LLE, and Laplacian Eigenmaps (top three rows in Figure 4). In terms of the top 20 outliers found rowwise, variable bandwidth could find most outliers lying on the left area of the embedding in ISOMAP and UMAP, and both methods in LLE embedding could find the outliers in the outer area, but for the other methods, both variable and fixed bandwidth are not detect true outliers accurately. For t-SNE and UMAP embedding, the embedding structure is highly distorted and the points are clustered together in a discontinuous way, which is also shown in the clustered outliers.

To further compare the accuracy of the estimated densities for all data points, we calculate the correlation between the rank of true densities and the estimated densities and present in Table 1. It can be seen that the rank correlation of our proposed method with variable bandwidth is higher for

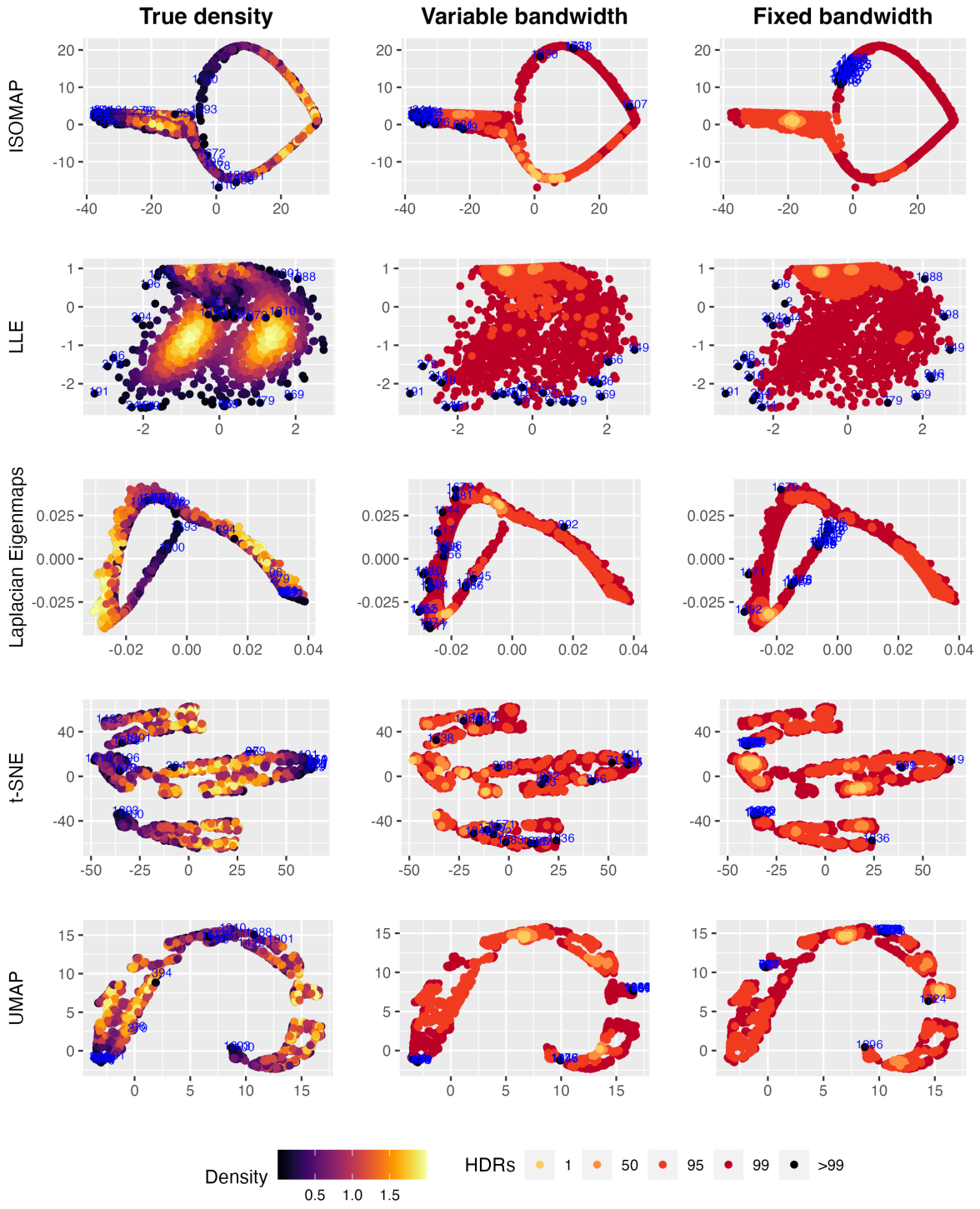

Figure 4: *Highest density region plots of five manifold learning embeddings of the swiss roll data. Colors are indicating densities from left: true densities from the Gaussian mixture model; middle: KDE with Riemannian matrix as variable bandwidth; and right: KDE with fixed bandwidth. Variable KDE preforms better in finding kernel structures with ISOMAP, LLE, and Laplacian Eigenmaps, and in locating outliers with ISOMAP and LLE. Both methods are not detecting outliers accurately when using t-SNE and UMAP.*

Table 2: *Correlation between true density and estimated density for four manifold learning embeddings.*

	ISOMAP	LLE	Laplacian.Eigenmaps	t.SNE	UMAP
Variable bandwidth	**0.899**	0.385	0.620	0.259	**0.659**
Fixed bandwidth	0.626	**0.399**	**0.622**	**0.663**	0.653

LLE and UMAP, although the correlation for UMAP is very close to zero. The highest correlation comes from our method in LLE embedding, which is mainly due to it is closest to rectangular structure of the meta data shown in Figure 2. For Laplacian Eigenmaps, our method has wrongly estimated the left area with lower densities even though their true densities are the very high in yellow, leading to a negative correlation. The negative correlation would occur typically when the highest or lowest true density areas are not well estimated. As for the estimates in highly distorted embedding, including ISOMAP, t-SNE, and UMAP, the rank correlations are quite low. In conclusion, our proposed method could improve the kernel density estimate of manifold learning embedding by considering the distortion using the Riemannian metric. However, if the distortion is too severe, eg. ISOMAP, or when the embedding are discontinuous, eg. t-SNE and UMAP, the density estimates are not as reliable for outlier detection.

3.1.2 Twin peaks mapping

For comparison, we use the same meta data in Figure 2 with different mapping function, twin peak mapping in Equation (3), with the corresponding 3-D data shown in the right plot of Figure 3.

$$\begin{cases} X = X_1, \\ Y = X_2, \\ Z = \sin(\pi X_1)\tanh(3X_2). \end{cases} \tag{3}$$

Similar to Figure 4, four manifold learning embeddings are obtained and used to detect outliers with two bandwidth selection methods in Figure 5. [ANALYSE TWIN PEAK PLOT]

With severe distortion in the t-SNE and UMAP embeddings, both variable and fixed bandwidth methods can not produce accurate density estimates. It would then be very difficult to uncover the Gaussian mixture model structure and

3.2 5-D data embedding in 100-D space

For the high-dimensional case, we generate the meta data from a 5-dimensional semi-hypersphere. First, we simulate $N = 2000$ points from a 4-dimensional Gaussian mixture model with two mixture components, $\mathrm{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathrm{N}(\boldsymbol{\mu}_2, \boldsymbol{\sigma}_2)$, where $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = (0,0,0,0)'$, $\boldsymbol{\Sigma}_1 = diag(1,1,1,1)$, and $\boldsymbol{\Sigma}_1 = diag(25,25,25,25)$. In order to manually generate anomalies, the mixture proportions are set as $\pi_1 = 0.99$ and $\pi_2 = 0.01$. The fifth dimension is calculated to satisfy $x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = r^2$

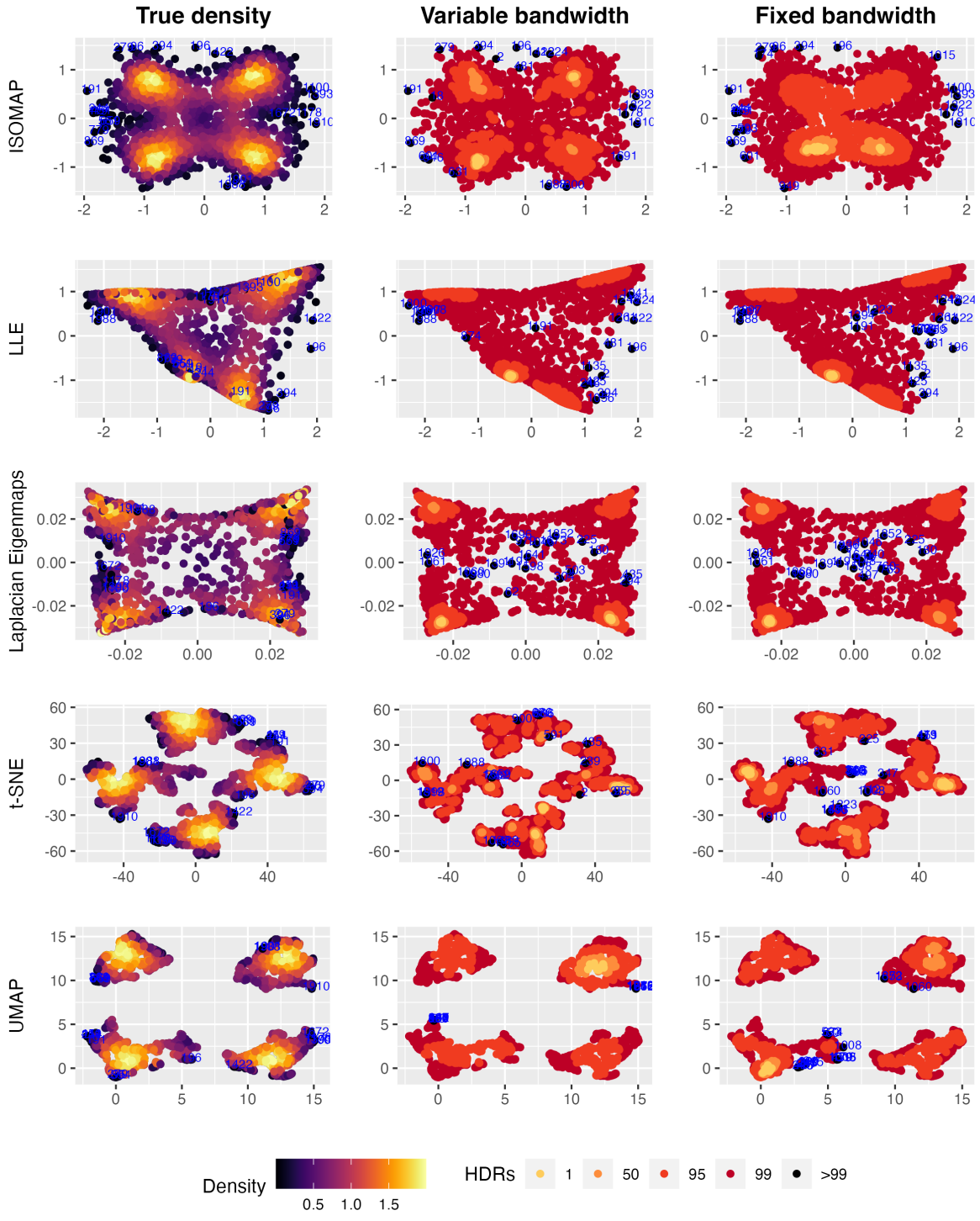

Figure 5: *Highest density region plots of four manifold learning embeddings of the twin peak data. Variable KDE preforms better in finding kernel structures with ISOMAP and LLE, and in locating outliers with t-SNE and UMAP.*

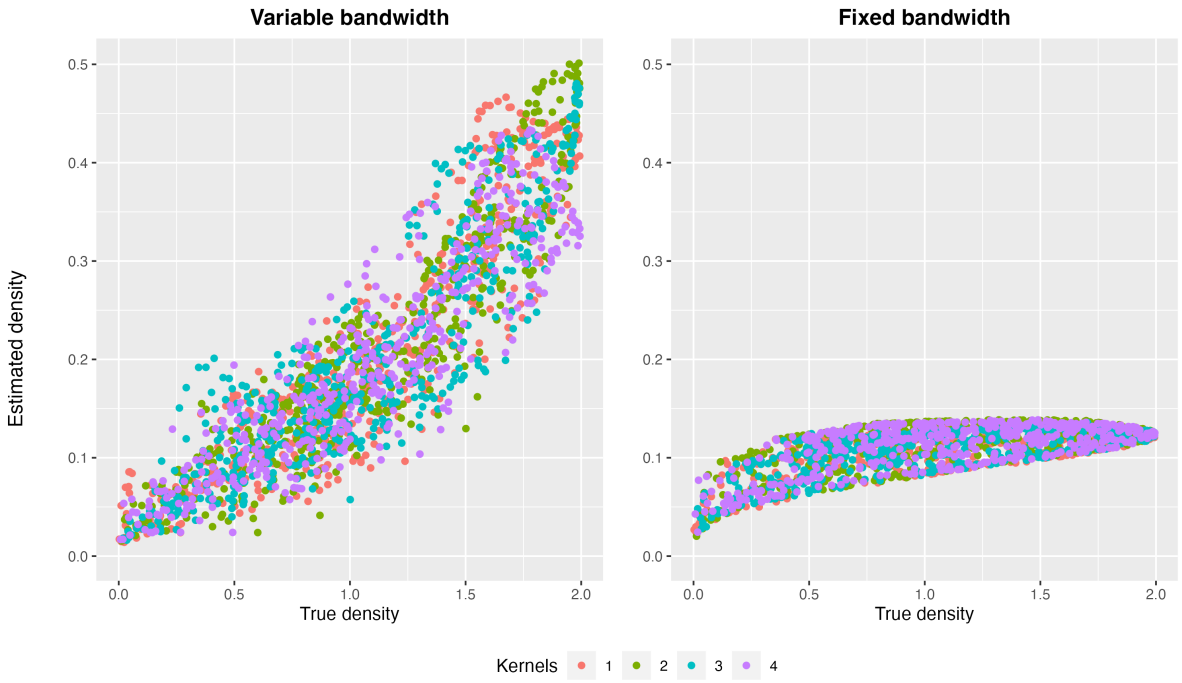

Figure 6: *Scatterplot of true density and estimated density of ISOMAP embedding for KDE with both variable and fixed bandwidth. The four colors represents the four gaussian kernels in the 2-D meta data.*

Table 3: *Correlation between true density and estimated density for four manifold learning embeddings.*

	ISOMAP	LLE	UMAP
Variable bandwidth	**0.921**	**0.981**	**-0.130**
Fixed bandwidth	0.806	0.940	-0.341

where r is set as 15. Similarly, the Gaussian mixture densities could be calculated using Equation (1) as the true density of the 5-d meta data. Figure 7 shows a scatterplot when animating a 5-D tour path with the R package *tourr* [REFERENCE]. Then we initial the other 95 dimensions in the high-dimensional space as zero columns and further rotate the 100-dimensional data of size N (denote the transpose of the data matrix as X) to get rid of the zeros so that it could be passed to the manifold learning algorithms. The rotation matrix is derived from the QR decomposition of a 100×100 matrix A with all components randomly generated from a uniform distribution $\mathcal{U}(0,1)$. For any real matrix A of dimension $p \times q$, the QR decomposition could decompose the matrix into the multiplication of two matrix Q and R so that $A = QR$, where the dimension of Q is a matrix with unit norm orthogonal vectors, $Q'Q = I$, and R is an upper triangular matrix. Matrix Q is used as the rotation matrix and it satisfies $X'X = (QX)'(QX)$, where the rotated data matrix $(QX)'$ is now the input for the manifold learning algorithms.

Similar to Figure 4, four manifold learning embeddings are obtained and used to detect outliers with two bandwidth selection methods in Figure 5. [ANALYSE TWIN PEAK PLOT]

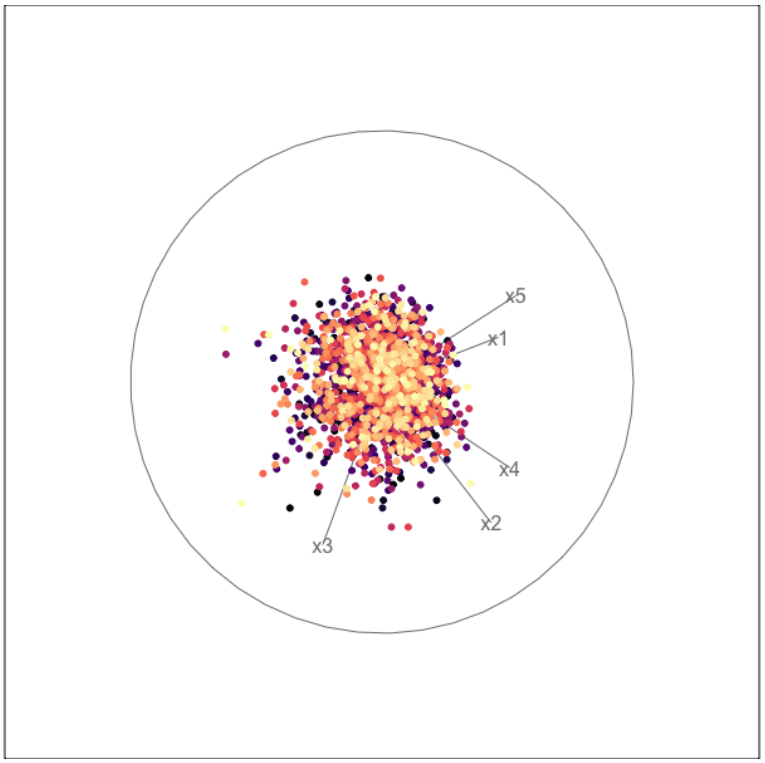

Figure 7: *Scatterplot display of the animation of a 5-D tour path.*

4 Application

4.1 Victoria smart meter dataset

[TODO]

Next, we consider smart-meter data for residential and non-profiled meter consumers, collected in the *CER Smart Metering Project - Electricity Customer Behaviour Trial, 2009-2010* in Ireland (**cer2012-data**). Electricity smart meters record consumption, on a near real-time basis, at the level of individual commercial and residential properties. The CER dataset[2] does not include energy for heating systems since it is either metered separately, or households use a different source of energy, such as oil or gas. In this study, the installed cooling systems are also not reported.

We use measurements of half-hourly electricity consumption gathered from 3,639 residential consumers over 535 consecutive days. Every meter provides electricity consumption between 14 July 2009 and 31 December 2010. Demand data from two smart meters (ID 1003 and 1539) are shown in **??** as time series plots. It is obvious that these meters have relatively different patterns. Meter 1539 (bottom of **??**) has a period of around 150 days with lower (approximately half) the electricity usage of remaining days and is otherwise relatively stable. In contrast, Meter 1003 (top of **??**) exhibits regular spikes on weekends.

[2] accessed via the Irish Social Science Data Archive - www.ucd.ie/issda.

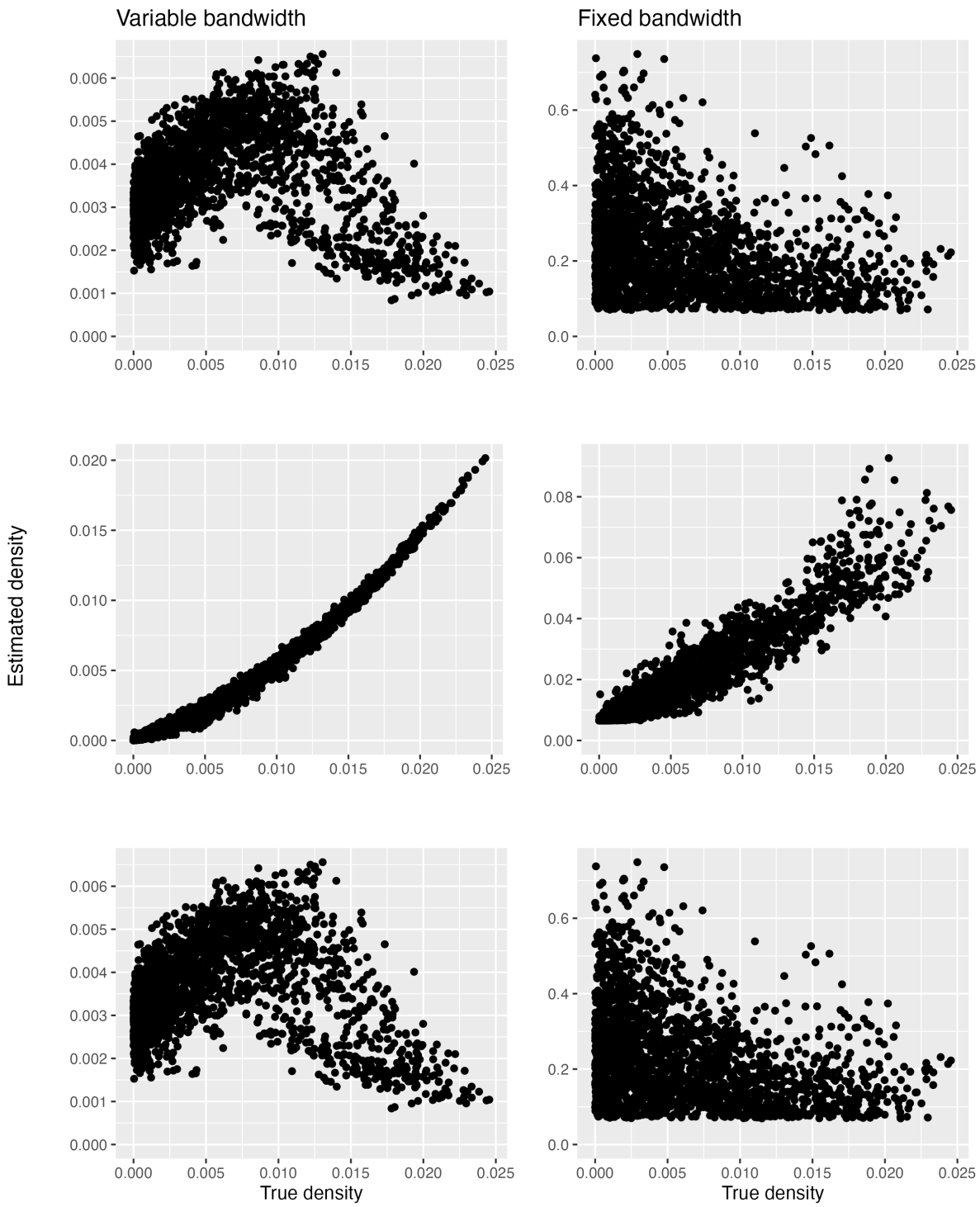

Figure 8: *Scatterplot of true density and estimated density of different embeddings for KDE with both variable and fixed bandwidth. Top: ISOMAP; middle: LLE; bottom: UMAP.*

For electricity demand data, one particular problem of interest is to visualize and identify households or periods of the week with anomalous usage patterns. For this reason, the object of interest in comparing households or periods of the week is the distribution of electricity demand rather than the raw data itself (Hyndman, Liu & Pinson 2018). An additional advantage of looking at distributions rather than raw data is that it provides a convenient mechanism for handling missing data which can be a significant problem in smart meter data applications. In the rest of this section, we first describe how discrete approximations to the distributions of interest are computed. We then approximate the distances between vectors containing probabilities with the consistent estimator in **??**, which allows us to apply manifold learning techniques on statistical manifolds.

Let $d_{i,t}$ denote the electricity demand for observation i and for time period t (subsequently we will see that i can either index the household, time of the week, or both, while t may index the week or half-hour period). The objective is to estimate the distances between the distribution of electricity demand for observation i, F_i, over time. Since the raw data could be thought of as samples generated from the true electricity usage distributions for different households and different periods, we could apply the consistent Hellinger distance estimator derived in **??** to the smart meter data. The first step is to group all data together regardless of i and t, and take the data as samples from the reference probability distribution R. The sample size $r = 93,616,914$. Next, we construct $T_r = 100$ intervals with about 1% samples from R within each interval (except for the last interval). The segments $\{I_g^r\}_{g=1,2,\ldots,100}$ can then be used to construct discrete distribution approximation to F_i by counting the ratio of samples points from F_i fallen into segment I_g^r. This is found by computing $\pi_{i,g} = (1/m)\sum_t I(d_{i,t} \in I_g^r)$ where m is the sample size from F_i. Then the Hellinger distance between F_i and F_j could be estimated using Equation (**??**). In this way, we make it possible to search nearest neighbors more efficiently using ANN methods and further apply manifold learning algorithms.

5 Conclusion

Summary of the contribution and conclusion

Details about the process

Shortcoming and other references for future direction (improvement on density estimation, other kernels, densities on edges, bad embeddings with inaccurate distortions, multi-dimensional kernel density estimate).

Discuss simulation with non-uniform meta data.

References

Goldberg, Y, A Zakai, D Kushnir & Y Ritov (2008). Manifold Learning: The Price of Normalization. *J. Mach. Learn. Res.* **9**(Aug), 1909–1939.

Hyndman, RJ, X Liu & P Pinson (2018). Visualizing big energy data: Solutions for this crucial component of data analysis. *IEEE Power Energ. Mag.*

Hyndman, RJ (1996). Computing and Graphing Highest Density Regions. *Am. Stat.* **50**(2), 120–126.

McQueen, J, M Meilă, J VanderPlas & Z Zhang (2016). Megaman: Scalable Manifold Learning in Python. *J. Mach. Learn. Res.* **17**(148), 1–5.

Perrault-Joncas, D & M Meila (2013). Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. arXiv: 1305.7255 [stat.ML].

Zhou, X & M Belkin (2011). Semi-supervised Learning by Higher Order Regularization. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Vol. 15. Proceedings of Machine Learning Research. JMLR Workshop and Conference Proceedings, pp.892–900.