Characterising HPD set topology

Qi Chen
Department of Statistics
University of Oxford
qi.chen@ox.ac.uk

October 6, 2024

1 Introduction(Quoted Yige Geng)

As the dimension becomes larger, it is challenging to estimate the credible set of the joint highest posterior density of the multivariate posterior density. Tree based credible set estimation by Jeong Eun. Lee and Geoff K. Nicholls **?** use a consistent estimator to measure the symmetric difference between our credible set estimate and the true HPD set of the target density sample. The main change is that we avoid methods based on estimating the copula in $[0,1]^d$, which are used by those authors in their actual density estimation examples and replace it with a direct estimation of the density, limiting it to The "Truncation" set is automatically selected.
We adopt the method of Li et al. (2016) **?**. The goal of density estimation is to find the binary division of $\Omega(p) = \{\Lambda_1, \ldots, \Lambda_K\}$ into super-rectangular cells, so that the density $f(p)$ is well approximated by a constant in each cell. By affine transformation mapping, it finds a good partition $\Delta = \Delta_k, K_k = 1$ in $[0,1]^d$ which we map back as sets $\Lambda_k = s^{-1}(\Delta_k; \Omega(p)), k = 1, ..., K$. Step 5 of Algorithm 3 shows a good split which divided the i^{th} dimension into m_g equal-sized bins for leaf Δ_k. The split stops until $n_k < 3$ and $D^* < \tau \frac{N^{1/2}}{N_k}$. Finally $[0,1]^d$ will be partitioned into k connected hypercubes.

5: **for** each $\Delta_k = [\boldsymbol{a}^{(k)}, \boldsymbol{b}^{(k)}]$ in Δ **do**
6: Denote points in Δ_k by $\{\boldsymbol{s}^{(k,j)}\}_{j=1}^{n_k}$. Using cell boundaries $\boldsymbol{a}^{(k)} = (a_{k,1}, ..., a_{k,d})$ and $\boldsymbol{b}^{(k)} = (b_{k,1}, ..., b_{k,d})$, rescale with $\tilde{\boldsymbol{s}}^{(k,j)} = \left(\frac{s_1^{(k,j)} - a_{k,1}}{b_{k,1} - a_{k,1}}, ..., \frac{s_d^{(k,j)} - a_{k,d}}{b_{k,d} - a_{k,d}} \right), j = 1, ..., n_k$.
7: Calculate gaps $\{h_{i,s}\}_{i=1,...,d, l=1,...,m_g-1}$ in Δ_k and $D^*(\{\tilde{\boldsymbol{s}}^{(k,j)}\}_{j=1}^{n_k})$ (see Appendix).
8: **if** $n_k > 2$ and $D^*(\{\tilde{\boldsymbol{s}}^{(k,j)}\}_{j=1}^{n_k}) > \tau\sqrt{N}/n_k$ **then**

Figure 1: Step 5 of Algorithm 3

2 Preliminaries

In this section, we will define the sense of being adjacent, and will give two algorithms constructing the adjacency structure computationally given a list of cells that are disjoint in a suitable sense, which will be defined soon. We first define the fundamental object we are dealing with: hyper-rectangle.

Definition 2.1. ***(Hyper-rectangle)*** *A d-dimensional hyper-rectangle is defined by the product d of closed intervals*

$$X = \prod_{i=1,...,d} [x_i^-, x_i^+]$$

where $-\infty < x_i^- < x_i^+ < +\infty$.

All "cells" below are are referred to d-dimensional hyper-rectangles unless stated otherwise.

Definition 2.2. ***(Disjoint)*** *Two cells are said to be disjoint if their intersection is of dimension less than* d.

Remark 2.1. *We make this definition since an edge, a surface or a hyper-plane of dimension less than* d *could be shared by two cells. But they never "overlap" in the sense that their intersection is also a* d*-dim object.*

We make two definitions of being adjacent, the reason of doing this will be clarified in the next section.

Definition 2.3. ***(Strongly Adjacent)*** *Two disjoint cells are said to be strongly adjacent if their intersection has dimension exactly* $d-1$.

Definition 2.4. ***(Weakly Adjacent)*** *Two disjoint cells are said to be weakly adjacent if their intersection is non-empty.*

Actually we can expand this further to a so-called "m-adjacent" definition.

Definition 2.5. ***(M-adjacency)*** *Two disjoint cells are said to be* m*-adjacent for* m *an integer,* $0 \leq m \leq d-1$*, if their intersection is of dimension* m.

For example, strong adjacency in definition 2.3 could be also treated as $d-1$-adjacent.
The cells in **?** method are obtained from partitioning the space horizontally and vertically by a density estimation tree method suggested by **?**. This means we only need to check the intervals on each dimension.
Observe that if two cells $X^{(1)}$ and $X^{(2)}$ are weakly(or strongly) adjacent, then at least at some dimension i, $[x_i^{-,(1)}, x_i^{+,(1)}]$ meets $[x_i^{-,(2)}, x_i^{+,(2)}]$ at exactly one point. Put it formally,

$$\exists i : [x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] \in \{x_i^{-,(1)}, x_i^{+,(1)}\}$$

In the case the intersection is $x_i^{-,(1)}$, we must have $x_i^{-,(1)} = x_i^{+,(2)}$ (which could be imagined as $X^{(2)}$ is to the "left" of $X^{(1)}$) and similarly, if the intersection is $x_i^{+,(1)}$, we must have $x_i^{+,(1)} = x_i^{-,(2)}$ (i.e. $X^{(2)}$ is to the "right" of $X^{(1)}$). For a sufficient and necessary condition of being weakly adjacent, we give the following lemma.

Lemma 2.1. ***(Sufficient and Necessary Condition for Weakly Adjacent)***
Two disjoint cells $X^{(1)} = \prod_{i=1,...,d} [x_i^{-,(1)}, x_i^{+,(1)}]$ *and* $X^{(2)} = \prod_{i=1,...,d} [x_i^{-,(2)}, x_i^{+,(2)}]$ *are weakly adjacent if and only if* $[x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] \neq \varnothing$ *for* $i = 1, ..., d$.

Proof. Sufficiency follows immediately from the definition: If $[x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] = I_i$ where I_i is non-empty for all i, then we can choose some elements for each I_i. Take the product we have non-trivial intersection of the two cells. Necessity is also basically from the definition: If $\exists i : [x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] = \varnothing$, then for any point in the intersection of two cells, the ith coordinate must be empty, which leads to a contradiction.

□

Algorithm 1 Check for weak adjacency

procedure WEAKADJDIRECT($X^{(1)}, X^{(2)}$) ▷ Cells are disjoint and are both of dimension d.
  for $i = 1, ..., d$ **do**
    if interval $[x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] = \varnothing$ **then**
      return FALSE
    end if
  end for
  return TRUE
end procedure

lemma 2.1 leads to an easy algorithm to check if two cells are weakly adjacent. Given two disjoint cells, we check if the intersection of the two intervals is non-empty on every dimension.

algorithm 1 has a computational cost $O(d)$ for each pair of cells. That means given k cells, we can get all the adjacency relationships at a cost $O(d * k^2)$.

For strong adjacency, we require a bit more. Since the dimension of the intersection must be exactly $d-1$, only one dimension i is allowed to have $[x_i^{-,(1)}, x_i^{+,(1)}]$ meets $[x_i^{-,(2)}, x_i^{+,(2)}]$ at exactly one point, and intervals must overlap on all other dimensions. This gives us algorithm 2.

Algorithm 2 Check for strong adjacency

procedure STRONGADJDIRECT($X^{(1)}, X^{(2)}$) ▷ Cells are disjoint and are both of dimension d.
  $k = 0$
  for $i = 1, ..., d$ **do**
    if interval $[x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] = \varnothing$ or $k = 2$ **then**
      return FALSE
    else if $[x_i^{-,(1)}, x_i^{+,(1)}] \cap [x_i^{-,(2)}, x_i^{+,(2)}] \in \{x_i^{-,(1)}, x_i^{+,(1)}\}$ **then**
      $k = k + 1$
    end if
  end for
  return TRUE
end procedure

Remark 2.2. *An easy way to get the intersection of intervals is to directly compare the endpoints. This is plausible in our case since algorithm partitions the space by first creating a chessboard-like lattice then choosing where to partition at one of the splitting points. This means we can deliberately make all cutting points finite decimal number. In general, this is would be more difficult because of floating-point error.*

We give a second method to determine the weak(strong) adjacency. This method would require the record of how we get the cells from partitioning the space. Starting from an initial cell $X_1^{(1)} = \prod_{i=1,...,d} [x_i^-, x_i^+]$, the makes the split by first choosing a dimension j and split at some point along this dimension. Suppose the splitting point is c_j, then we end up with two cells $X_1^{(1)} = \prod_{i=1,...,d, i\neq j} [x_i^-, x_i^+] \times [x_j^-, c_j]$ and $X_2^{(2)} = \prod_{i=1,...,d, i\neq j} [x_i^-, x_i^+] \times [c_j, x_j^+]$. We say $X_1^{(2)}$ is to the left of $X_2^{(2)}$ in the sense that $x_j^- < c_j < x_j^+$. In the next loop, both $X_1^{(2)}$ and $X_2^{(2)}$ are visited, and the algorithm will then partition one of these into two cells. This process could be

stored in a binary tree since we always partition a leaf node into two cells. Interested readers are referred to the appendix for more details.

Algorithm 3 Tree-based check for adjacency

procedure WEAKADJTREE($X^{(1)}, X^{(2)}, T$) ▷ Cells are disjoint and are both of dimension d ▷ T is a binary tree, each node consists of three elements: s the cutting dimension, c the cutting coordinate on this dimension, z if this cell is being partitioned to the left, then $z = L$, otherwise $z = R$
  $n_0 = X^{(1)}$
  $m_0 = X^{(2)}$
  Backtrack n_0, m_0 to the closest common ancestor in the tree T. Let the back tracing chains be $n_0 - n_1 - \cdots - n_p - l$ and $m_0 - m_1 - \cdots - m_q - l$, where l is the closest common ancestor.
  for $i = p, \ldots, 1$ **do**
    if $n_i.d = l.d$ **and** $n_i.z = L$ **then**
      return FALSE
    else if $(n_i.z = L$ **and** $n_i.c < x^{-,(2)}_{(n_i.d)})$ **or** $(n_i.z = R$ **and** $n_i.c > x^{+,(2)}_{(n_i.d)})$ **then**
      return FALSE
    end if
  end for
  for $i = q, \ldots, 1$ **do**
    if $m_i.d = l.d$ **and** $m_i.z = L$ **then**
      return FALSE
    else if $(m_i.z = L$ **and** $m_i.c < x^{-,(1)}_{(n_i.d)})$ **or** $(m_i.z = R$ **and** $m_i.c > x^{+,(1)}_{(m_i.d)})$ **then**
      return FALSE
    end if
  end for
end procedure

The computational cost of algorithm 3 is $O(\log k)$ for any pair of cells given that we partition the whole space into k cells. That means we can construct all the adjacency relationships by a cost $O(k^2 * \log k)$.

Remark 2.3. *For a test of strong adjacency based on tree data structure, we only need to restrict the number of times allowed that the intervals meet at the endpoints. This could be done easily similar to the difference in algorithm 1 and algorithm 2 by a variable record the endpoints-meet cases.*

Remark 2.4. *Notice that algorithm 3 does not strictly dominate algorithm 1, they are equally time-consuming when the number of cells k equals to 2^d where d is the dimension of the data. But algorithm 3 would be more space-consuming since it will require a tree of space $O(\log k)$.*

3 Core theorem and erosion

3.1 Topology

For the topology of the HPD cells, what we want is to first remove some of the HPD cells while the topological properties don't change, in the sense that the union of HPD cells before removing and after removing are homotopic equivalence. We notice that if we look locally at a d-cell,

its boundary is a collection of surfaces with dimension $d' \leq d-1$ being split into two classes: The part where the cell is adjacent to a HPD cell, and the part where the cell is adjacent to a complement cell. Notice that these two classes are not disjoint, it is easy to imagine in 2D, an vertex might be shared by multiple cells (instead of just two cells) and so it may both be adjacent to a HPD cell and a complement cell. Similar case happens in higher dimension. But the $d-1$ surfaces in this collection is well partitioned by the two classes.

Definition 3.1. ***(D-connected)***

Notice that in the space S, we only have two classes of cells, the HPD cells, and the complement of the HPD cells. We will assume the space is partitioned into k disjoint cells $X^{(1)}, \ldots, X^{(n)}$, and the HPD cells are contained in a set $H = \bigcup_{i \in I} X^{(i)}$, where $I \subset \{1, \ldots, n\}$. $C = S \backslash H$ as the complement.

Definition 3.2. ***(Cells at the boundary)*** *We say $X^{(i)}$ is at the boundary of the HPD set if $X^{(i)} \cap C$ is of dimension $d-1$ for $i \in I$, and $X^{(i)} \cap (H \backslash X^{(i)})$ is of dimension $d-1$.*

Definition 3.3. ***(Neighbours)*** *The neighbours of a cell X is the union of cells such that they are weakly adjacent to X.*

We construct the adjacency graph via the algorithm 1 and algorithm 2 in the last section. The graph consists of k vertices representing the k cells. There are two types of edges, which are the SA edges representing the two cells are strongly adjacent to each other and the WA edges representing a weak adjacency but not strong adjacency, since the case of being strong adjacency is included in the definition of weak adjacency.

Definition 3.4. ***(Removable)*** *Denote the neighbours of $X^{(i)}$ by N. We say a cell $X^{(i)}$ is removable where $i \in I$ if $X^{(i)}$ is at the boundary, the induced sub-graph by vertices $N \cap H$ is connected by SA edges, and the induced sub-graph by vertices $N \cap C$ is connected by NA edges.*

Conjecture 3.1. *H is homotopic equivalent to $H \backslash X^{(i)}$ if $X^{(i)}$ is removable.*

Proof. □

There are various ways to implementing definition 3.4. Below we give a relatively straight way in algorithm 4.

4 Experiments

In this section we run algorithm 4 on HPD sets estimated by . We deliberately make the distribution in section 4.1 and section 4.2 so that we know what the topological structure is likely to be beforehand. Therefore we can compare the experiment results with our prior knowledge. In section 4.3 we use real-life data.

4.1 Two dimensional data

In this subsection we make two distributions. In section 4.1.1 we simulate points from a 2-dim multivariate normal distribution, which we are confident that the topological structure of the HPD sets estimated is likely to be a closed and connected region. In section 4.1.2 we simulate points on a double-torus on a 2-dim plane, the details will be given later.

Algorithm 4 Erosion algorithm

procedure EROSION(H, C, G) ▷ H is the set of HPD cells, C is the complement and G is the adjacency graph
    B = cells at the boundary of the HPD set
    while true **do**
        for X in B **do**
            N = neighbours of X
            if $N \cap H$ is connected by SA edges **and** $N \cap C$ is connected by SA edges **then** ▷ definition 3.4
                $C = C \cup X$
                $H = H \backslash X$
                Update the cells at the boundary from $N \cap H$
            end if
        end for
    end while
end procedure

4.1.1 Multivariate normal data

Here we simulate 2-dim multivariate normal distribution twice, the plots are given in fig. 2 and fig. 3. The reason of doing this is that the HPD set estimation is sometimes unstable. In the first experiment given in fig. 2, estimation fails to capture a cell on the right up corner, so that we have a 'hole' in the region. This is captured by algorithm 4 successfully and we can see that there's a cycle in the induced sub-graph. In the second experiment given in fig. 3, we see that the estimated HPD cells is indeed a connected region. This is also reflected in the cells remaining after erosion: we only have one cell remaining, indicating we have a only one connected part in the original HPD sets and the topological property is rather trivial.

4.1.2 Double-torus data

In this subsection we want algorithm 4 to capture a "eight-shape" structure in the data. To make sure the HPD cells are also distributed like two rings, we make the points denser around the "eight-shape" and sparser when the distance gets larger. In the following example we simulate points around a circle $(x-1)^2 + y^2 = 0.55$ and another circle $(x+1)^2 + y^2 = 0.55$. We first simulate points around $x^2 + y^2 = 0.55$. This is done by first simulating a normal distribution $r \sim N(0, 0.55/4^2)$, this would be the distance from the circle, and if it is less than 0 means the point is shifted towards the origin, while if it is positive the point is shifted away the origin. Then an angle is simulated uniformly on $[0, 2\pi]$. This gives the coordinate $((r+0.55)*cos(\theta), (r+0.55)*sin(\theta)$ on the xy plane. Finally we make a choice to whether move the point to the left or right horizontally by 0.55. This is done by a $Bernoulli(0.5)$ random number.

Remark 4.1. *The points are not uniformly dense with a fix distance inside and outside the circle. It is actually denser when it comes inside the circle. This is would not influence anything related to our result.*

We can see from fig. 4 that the structure of two rings sticking together is successfully captured by our erosion method. This is clear from sub-graph in the original adjacent graph induced by the cells remaining.

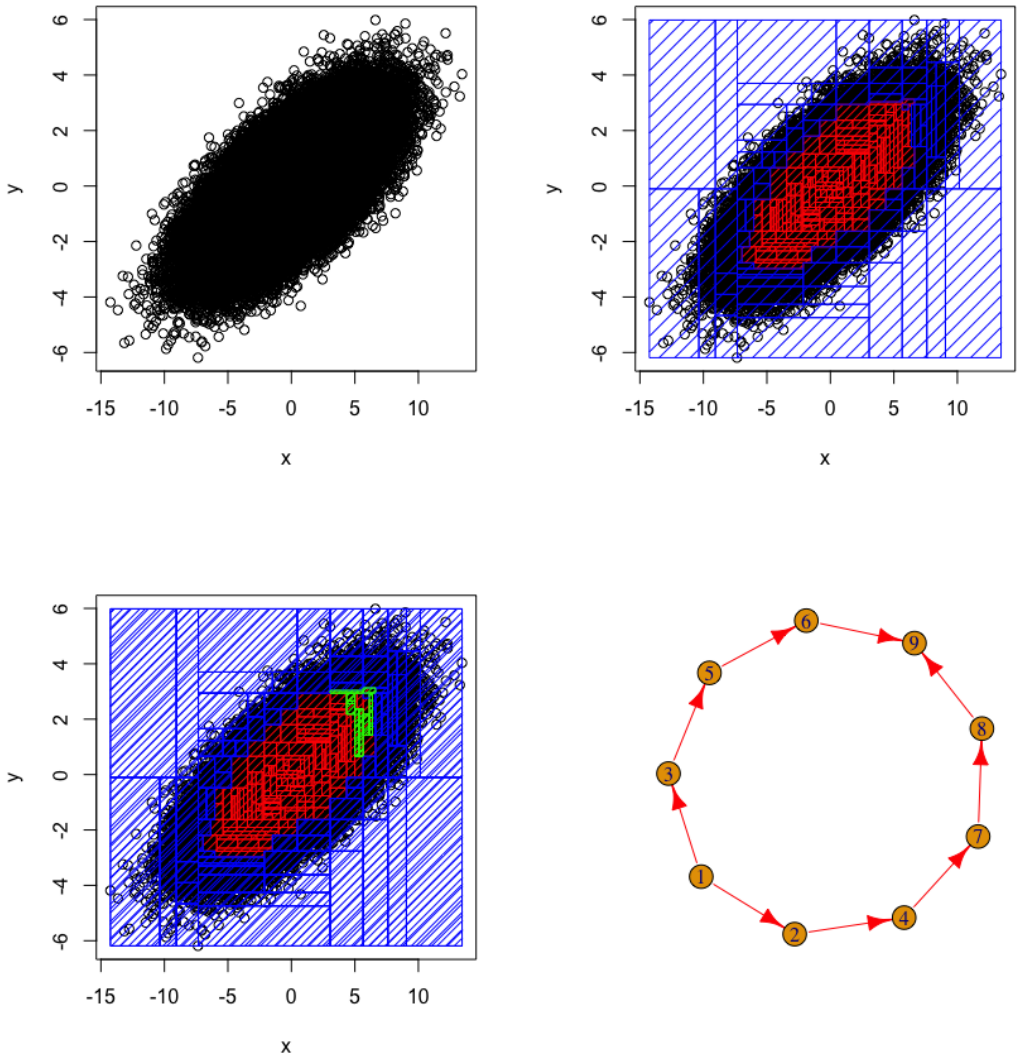

Figure 2: First experiment of 100000 samples from the 2-dim multivariate normal distribution (top row, i) and the HPD set is estimated colored in red while the complement to HPD set is colored in blue(top row, ii); the HPD cells after erosion are colored in green(bottom row, iii) and the induced sub-graph of remaining cells from the adjacency relationship graph.

4.2 Three dimensional data

4.3 Radiocarbon-dating data

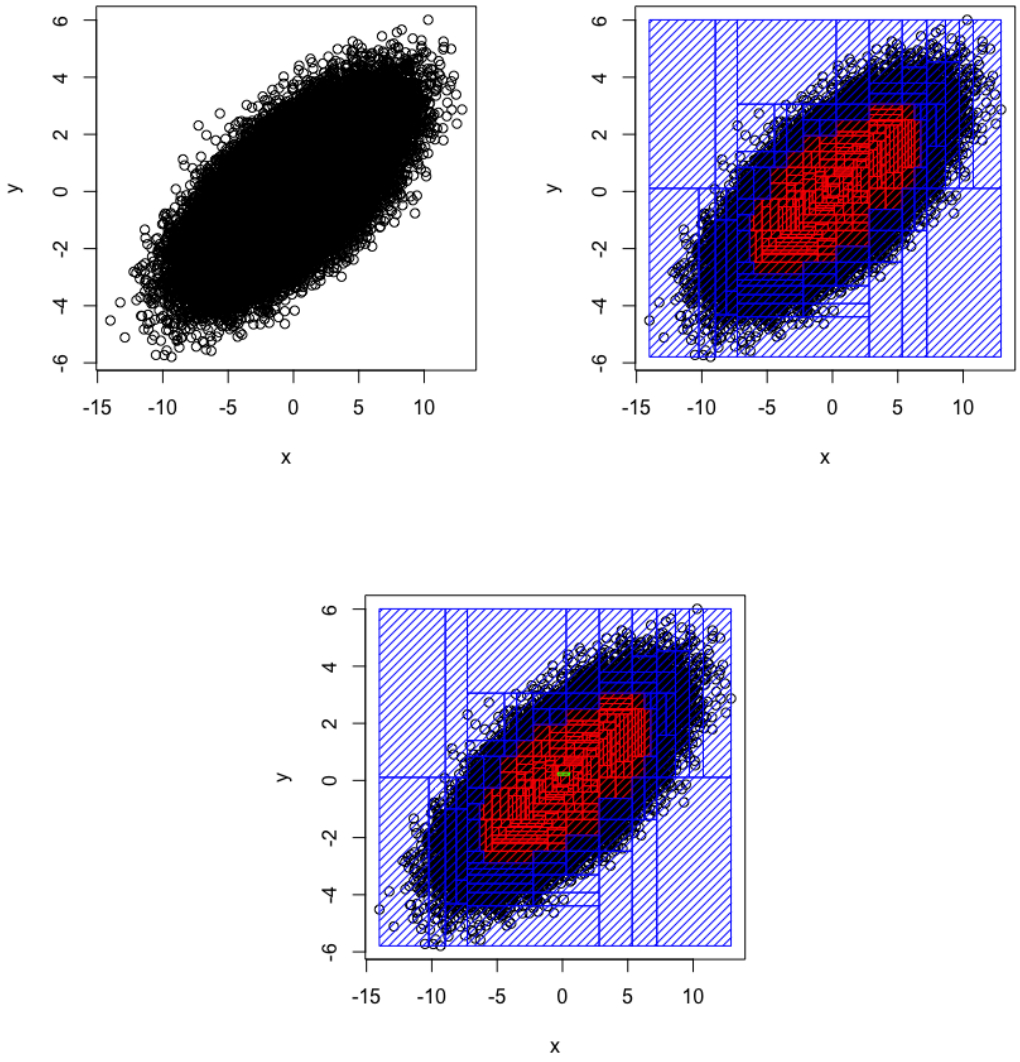

Figure 3: Second experiment of 100000 samples from the 2-dim multivariate normal distribution (top row, i) and the HPD set is estimated colored in red while the complement to HPD set is colored in blue(top row, ii); the HPD cells after erosion are colored in green(bottom row).

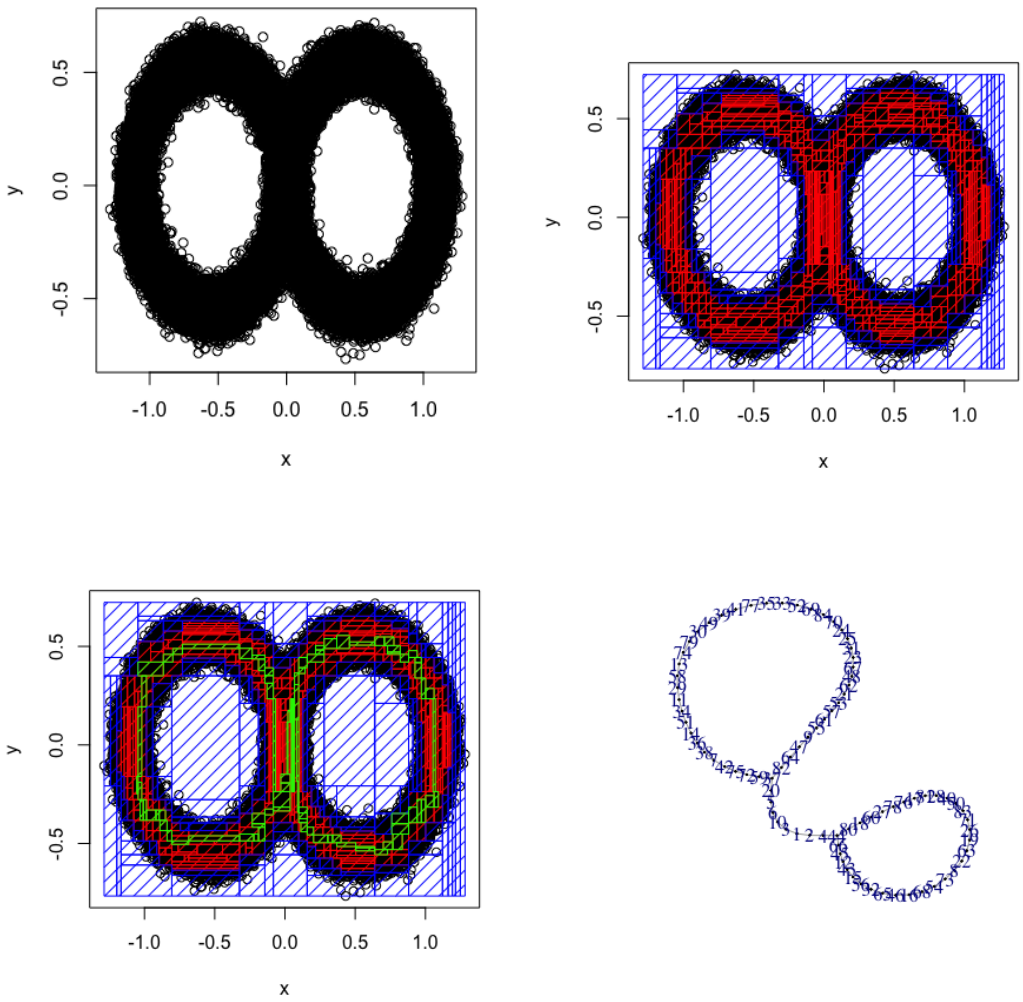

Figure 4: 100000 samples from the eight-shape distribution (top row, i) and the HPD set is estimated colored in red while the complement to HPD set is colored in blue(top row, ii); the HPD cells after erosion are colored in green(bottom row, iii) and the induced sub-graph of remaining cells from the adjacency relationship graph.