Scalable Minimum Distance Estimator for L1-Density Estimation with Universal Performance Guarantees

Raazesh Sainudiin, Axel Sandstedt, Johannes Graner, Tilo Wiklund,
Pauliina Ilmonen, Lauri Viitasaari and Warwick Tucker

Monday 18, December, 2023, Lisbon, Portugal, IMS Intl. Conference in Statistics & Data Science

Partly supported by Combient Mix's Summer Internship to Axel and a grant from Wallenberg AI, Autonomous Systems and Software Program funded by Knut and Alice Wallenberg Foundation to Raaz

Outline

- What is this all about & why do we care?
- What do we provide and the sources of ideas and tools?
- What is it good for?
- How do we do it?

What is this all about & why do we care?

Probability Model

there are n samples in d dimensional reals $\mathbb{R}^d$

samples are independently generated from an unknown density f

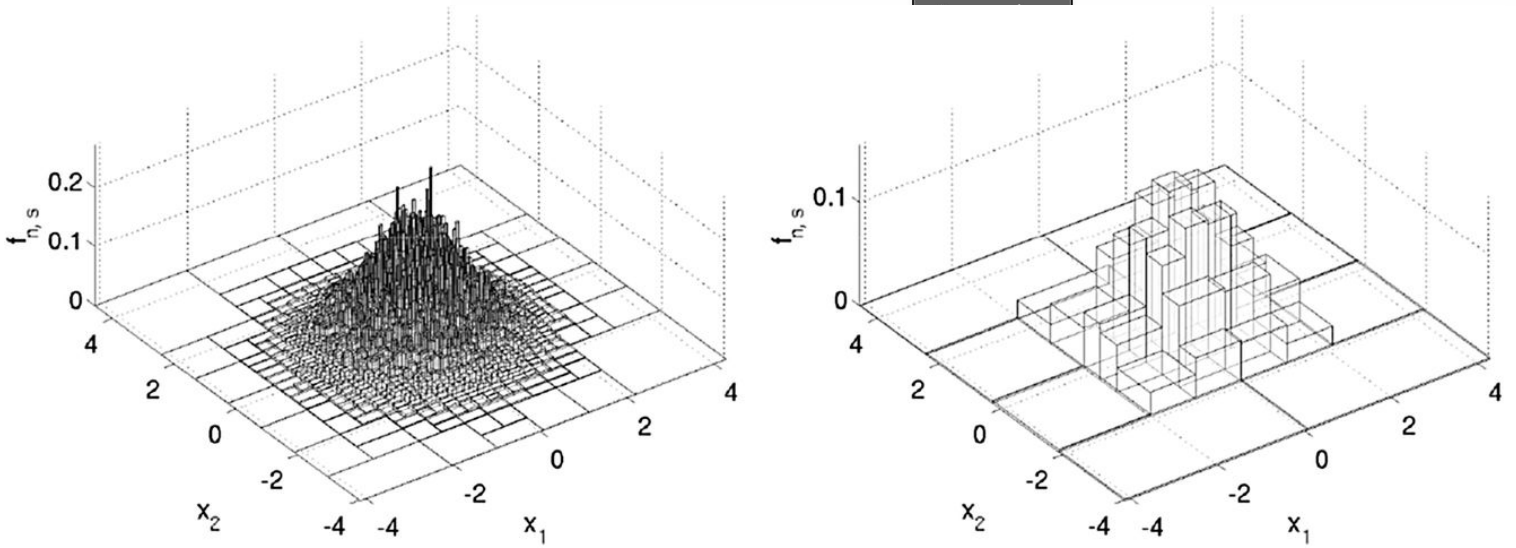
that belongs to the space of all possible densities, i.e., L_1

$$\mathbb{R}^d \ni X_1, X_2, \ldots, X_n \overset{iid}{\sim} f$$

$$f \in L_1 := \{g : \int |g| < \infty\}$$

For given sample size n **construct** an *optimally smoothed* Density Estimate (multidimensional histogram for eg.)

$$f_n(x; X_1, \ldots, X_n) : \mathbb{R}^d \times \left(\mathbb{R}^d\right)^n \to \mathbb{R}$$



Henry Scheffé's Identity

1.1 The L_1 setting

We start by bringing forth some definitions and results found in [1] which we shall need later. Consider a $\mathcal{B}(\mathbb{R})$-measurable random variable $\mathbf{X} : \Omega \to \mathbb{R}^d$ with density f and suppose that $g : \mathbb{R}^d \to \mathbb{R}$ is some other density. The L_1 distance between f and g may be defined as

$$\int_{\mathbb{R}^d} |f - g|.$$

The distance has the nice property of being easily interpreted, and is connected to the total variation by Scheffé's Identity:

$$\sup_{A \in \mathcal{B}(\mathbb{R}^d)} \left| \int_A f - \int_A g \right| = \frac{1}{2} \int_{\mathbb{R}^d} |f - g|.$$

Henry Scheffé's Identity

1.1 The L_1 setting

We start by bringing forth some definitions and results found in [1] which we shall need later. Consider a $\mathcal{B}(\mathbb{R})$-measurable random variable $\mathbf{X} : \Omega \to \mathbb{R}^d$ with density f and suppose that $g : \mathbb{R}^d \to \mathbb{R}$ is some other density. The L_1 distance between f and g may be defined as

$$\int_{\mathbb{R}^d} |f - g|.$$

The distance has the nice property of being easily interpreted, and is connected to the total variation by Scheffé's Identity:

$$\sup_{A \in \mathcal{B}(\mathbb{R}^d)} \left| \int_A f - \int_A g \right| = \frac{1}{2} \int_{\mathbb{R}^d} |f - g|.$$

Henry Scheffé's Identity

1.1 The L_1 setting

We start by bringing forth some definitions and results found in [1] which we shall need later. Consider a $\mathcal{B}(\mathbb{R})$-measurable random variable $\mathbf{X} : \Omega \to \mathbb{R}^d$ with density f and suppose that $g : \mathbb{R}^d \to \mathbb{R}$ is some other density. The L_1 distance between f and g may be defined as

$$\int_{\mathbb{R}^d} |f - g|.$$

The distance has the nice property of being easily interpreted, and is connected to the total variation by Scheffé's Identity:

$$\sup_{A \in \mathcal{B}(\mathbb{R}^d)} \left| \int_A f - \int_A g \right| = \frac{1}{2} \int_{\mathbb{R}^d} |f - g|.$$

We want a Computationally Efficient Estimator with UPG

What is of interest to us is the construction of an efficient estimator f_n in the computational sense and deriving a universal upper bound on the expected L_1 distance to f. We say that the estimator f_n is *additive* if for some measurable function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, f_n can be written in the form

$$f_n = \frac{1}{n} \sum_{i=1}^{n} K(x; X_i).$$

Any such f_n is called *regular* if $\forall x : \mathbb{E}|K(x; X)| < \infty$. Consider a multivariate density histogram h_n with a finite number of non-intersecting and non-empty cells with finite volume $P_1, \ldots, P_m$. The density may be written as

$$h_n(x; X_1, \ldots, X_n) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mathbb{I}_{[x \in P_j]} \mathbb{I}_{[X_i \in P_j]}}{\text{volume}(P_j)} = \frac{1}{n} \sum_{i=1}^{n} K(x; X_i).$$

Want: A histogram estimate from data (empirical measure)

Since K is the sum of a finite number of measurable functions, we conclude that h_n is additive. Regularity follows from observing that for any $x \in \mathbb{R}^d$, the following holds:

$$|K(x;X)| \leq \max_{j \leq m} \frac{1}{\text{volume}(P_j)}.$$

For a i.i.d sample $(X_1, \ldots, X_n)$ we define the *empirical measure* μ_n as

$$\mu_n(A) := \mu_n(A; X_1, \ldots, X_n) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{[X_i \in A]},$$

Yatracos Class & Minimum Distance Estimate (MDE)

Consider two densities $f_{n,\theta}, f_{n,\omega} : \mathbb{R}^d \to \mathbb{R}$. We define a *Scheffé set* as the set

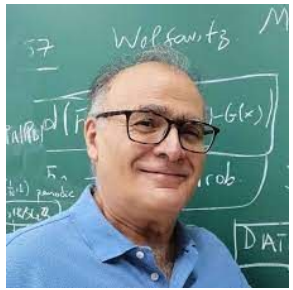
$$A(f_{n,\theta}, f_{n,\omega}) := \{x : f_{n,\theta}(x) > f_{n,\omega}(x)\}.$$

Now, suppose that we have a finite set of estimate indices $\theta \in \Theta$. The set of all distinct Scheffé sets of Θ is denoted by

$$\mathcal{A}_\Theta := \{A_{f_{n,\theta}, f_{n,\omega}} : \theta \neq \omega\}$$

and is known as the *Yatracos class* of Θ. Furthermore, assume that for any $\theta \in \Theta$, $f_{n,\theta}$ represents an estimate of some unknown but fixed density f. Define Δ_θ as the supremum of the absolute distance between $f_{n,\theta}$ and μ_n over $\mathcal{A}_\Theta$:

$$\Delta_\theta := \sup_{A \in \mathcal{A}_\Theta} \left| \int_A f_{n,\theta} - \mu_n(A) \right|.$$



The Minimum Distance Estimate (MDE)

The *minimum distance estimate* ψ_n (MDE) within the set of estimators in Θ generating $\mathcal{A}_\Theta$ is defined to be the estimate minimising the above distance:

$$\psi_n := f_k, \quad k = \min \operatorname*{argmin}_{i \in 1,\ldots,m} \Delta_i.$$

Shatter Coefficients and Vapnik-Chervonenkis Dimension

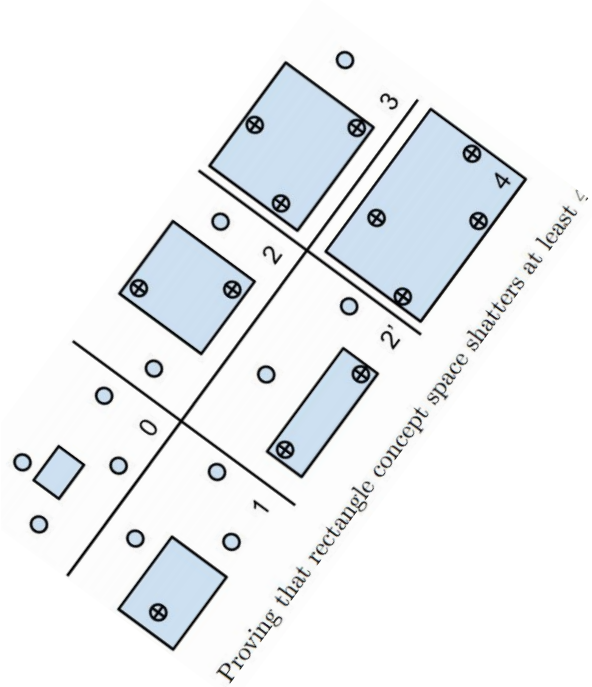
We shall need some more tools to formally deal with partitions of $\mathbb{R}^d$. Let $\mathcal{A}$ denote a class of subsets $A \subseteq \mathbb{R}^d$. We let $\mathcal{S}_{\mathcal{A}}(n)$ denote the *shattering coefficient* of $\mathcal{A}$ depending on n, and it is defined by

$$\mathcal{S}_{\mathcal{A}}(n) := \max_{(x_1,\ldots,x_n)\in(\mathbb{R}^d)^n} \left|\{A \cap \{x_1,\ldots,x_n\} : A \in \mathcal{A}\}\right|.$$

Intuitively, $\mathcal{S}_{\mathcal{A}}(n)$ describes the largest amount of unique ways the class could shatter or split a set of n points residing in $\mathbb{R}^d$. The *Vapnik-Chervonenkis dimension* $V_{\mathcal{A}}$ of $\mathcal{A}$ is defined as the largest integer n such that $\mathcal{S}_{\mathcal{A}}(n) = 2^n$. Thus $V_{\mathcal{A}}$ represents the largest number of points which $\mathcal{A}$ can fully shatter. Vapnik and Chervonenkis (1971) provided the following theorem which relates shattering coefficients to an i.i.d sample $(X_1,\ldots,X_n)$ coming from a distribution μ and its corresponding empirical measure:

([3], Theorem 12.5): Given a class of sets $\mathcal{A}$, a sample $(X_1,\ldots,X_n)$ with common probability distribution μ and any $\epsilon > 0$, the following inequality holds:

$$\mathbb{P}\left(\sup_{A\in\mathcal{A}}\left|\mu_n(A) - \mu(A)\right| > \epsilon\right) \leq 8S_{\mathcal{A}}(n)e^{-n\epsilon^2/32}.$$



Lugosi & Nobel (‘91) generazile VC dimension to Partitions

Let π denote a finite collection of non-intersecting $\mathcal{B}(\mathbb{R}^d)$-measurable subsets $A \subseteq \mathbb{R}^d$ such that $\bigcup_{A\in\pi} A = \mathbb{R}^d$. We are interested in the properties of certain families of partitions, and as such let $\mathcal{A}$ denote any family of possibly infinitely many partitions π. Furthermore, let

$$m(\mathcal{A}) := \sup_{\pi\in\mathcal{A}} |\pi|$$

denote the supremum over $\mathcal{A}$ with respect to the number of cells of any π. We define and measure the complexity of $\mathcal{A}$ by its ability to split a number of points; let $x_1, \ldots, x_n \in \mathbb{R}^d$ and $B = \{x_1, \ldots, x_n\}$. We define $\Delta(\mathcal{A}, B)$ as the number of unique ways $\mathcal{A}$'s partitions may split B:

$$\Delta(\mathcal{A}, B) := \Big|\{\{A_1 \cap B, \ldots, A_r \cap B\} : \pi = \{A_1, \ldots, A_r\} \in \mathcal{A}\}\Big|.$$

The *growth function* of $\mathcal{A}$ is defined as

$$\Delta^*(\mathcal{A}, n) := \max_{B=\{x_1,\ldots,x_n\}\subset\mathbb{R}^d} \Delta(\mathcal{A}, B).$$

Lugosi & Nobel (‘91) generazile VC dimension to Partitions

TODAY in Lisbon, Portugal and IRL “mathematical cuddle”

Concentration ineq: Empirical deviation from true measure

In the context of histogram estimators whose cells depend on the data, if we view the rules for constructing cells as fixed and $\mathcal{A}$ as the family of possible partitions π constructed using the rules, $\Delta^*(\mathcal{A}, n)$ measures in some sense the estimator's reliance on individual points versus a set of points, and is therefore of importance. This can in turn be related to the usual smoothing problem in which one chooses some bandwidth which affects how an estimator at any point $x \in \mathbb{R}^d$ relies on nearby sample points. The following lemma bridges the concept of growth functions and the empirical measure's performance over $\mathcal{A}$.

([2], Lemma 1): Let $\mathcal{A}$ be any family of partitions of $\mathbb{R}^d$. Let $\epsilon > 0$ and $n \geq 1$. Then the following holds:

$$\mathbb{P}\left(\sup_{\pi \in \mathcal{A}} \sum_{A \in \pi} \left| \mu_n(A) - \mu(A) \right| > \epsilon \right) \leq 4\Delta^*(\mathcal{A}, 2n) 2^{m(\mathcal{A})} e^{-n\epsilon^2/32}.$$

Almost sure convergence of any partitioning scheme

A corollary of this is that, for a given a sequence of i.i.d vectors $X_1, X_2, \ldots$ with common distribution μ and a sequence of partition families $\mathcal{A}_1, \mathcal{A}_2, \ldots$, if we put certain limitations on how quickly the two sequences $\{\Delta^*(\mathcal{A}_n, n)\}_{n=1}^{\infty}$ and $\{m(\mathcal{A}_n)\}_{n=1}^{\infty}$ are allowed to grow, then we can ensure almost sure convergence of the empirical measure μ_n to μ over $\mathcal{A}_n$:

([2], Corollary 1): Consider the setup described above. As $n \to \infty$, if $n^{-1}m(\mathcal{A}_n) \to 0$ and $n^{-1}\ln(\Delta^*(\mathcal{A}_n, n))) \to 0$, then it is true that

$$\mathbb{P}\left(\lim_{n\to\infty} \sup_{\pi\in\mathcal{A}} \sum_{A\in\pi} \left|\mu_n(A) - \mu(A)\right| = 0\right) = 1.$$

Regular Paving: For any sample size in any dimension d.

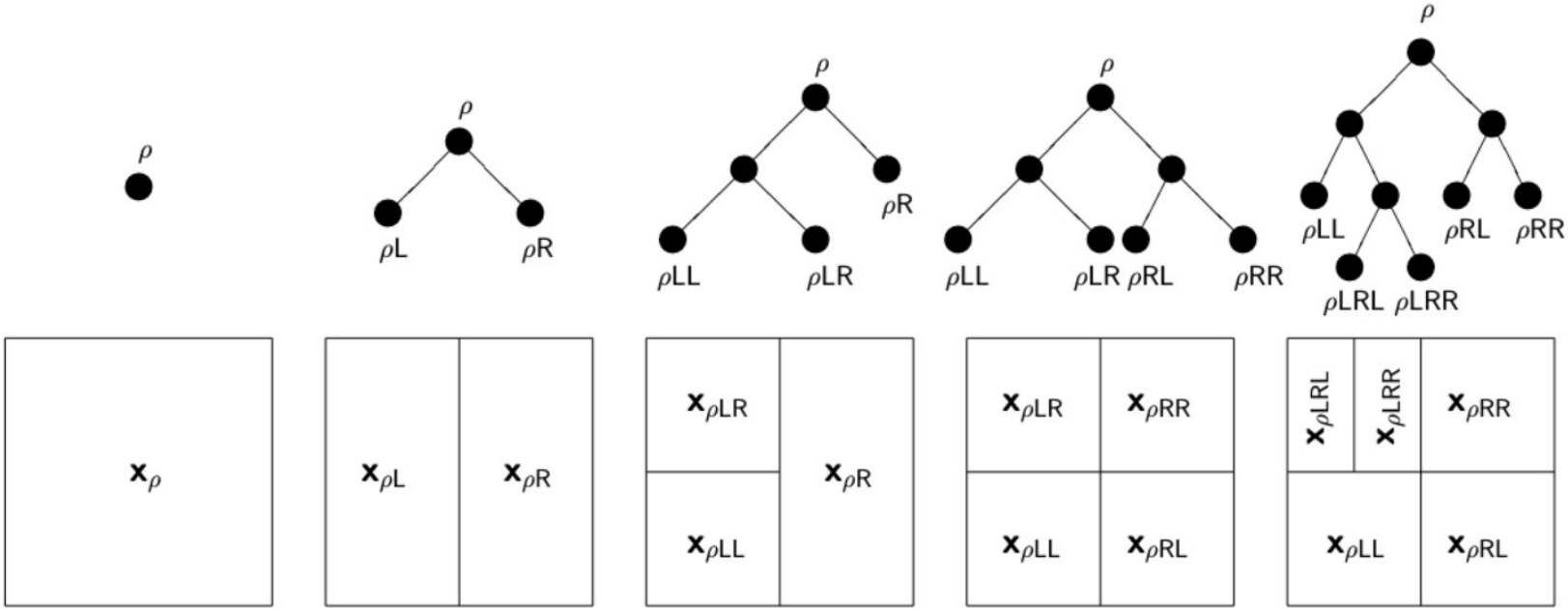
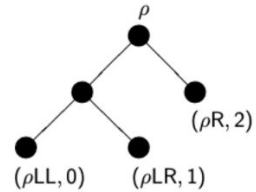


Figure: One possible sequence of splits achieving the rightmost RP.

Mapped Regular Paving

Suppose that ρ is some RP and let $\mathbb{Y}$ be some non-empty set.
A $\mathbb{Y}$-*mapped regular paving* is a map $f : \mathbb{V}(s) \to \mathbb{Y}$.



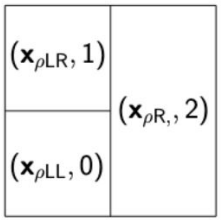
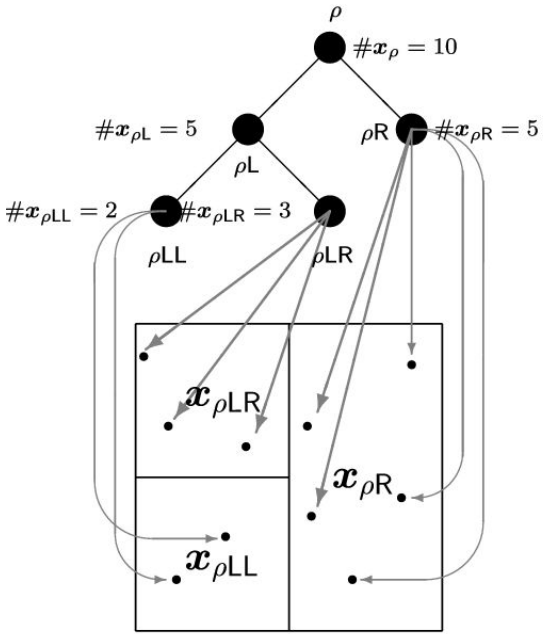
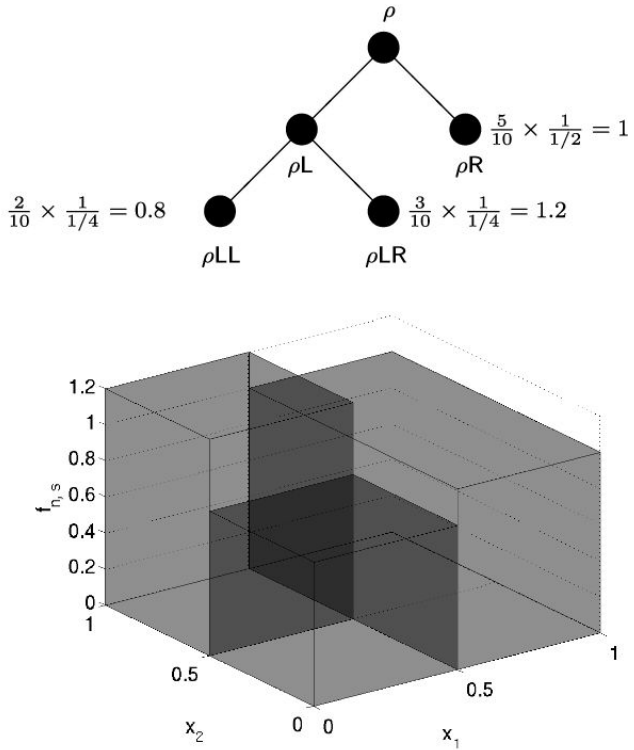


Figure: Simple $\mathbb{Z}$-MRP consisting of two splits.

Statistical Regular Paving Histogram

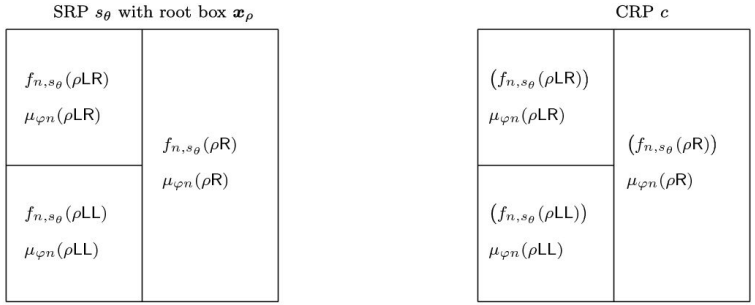


(a) An SRP tree and its constituents.

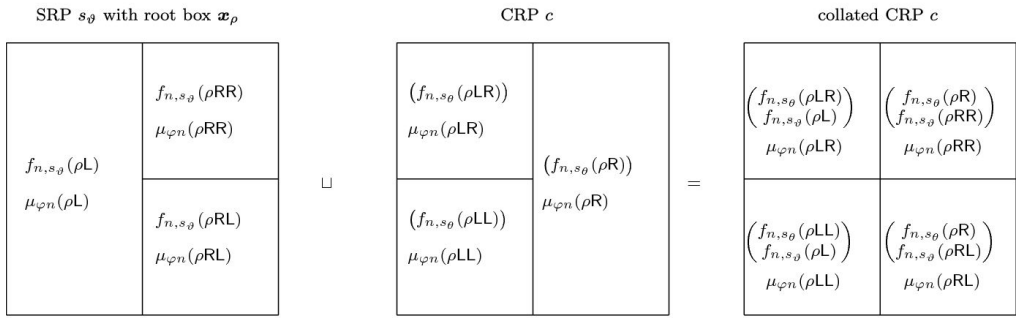


(b) An SRP histogram $f_{n,s}$.

Collator Regular Paving



(a) Make the SRP s_θ into a CRP c.



(b) Collate another SRP s_ϑ onto CRP c.

Sparse Regular Paving = Sparse Binary Tree

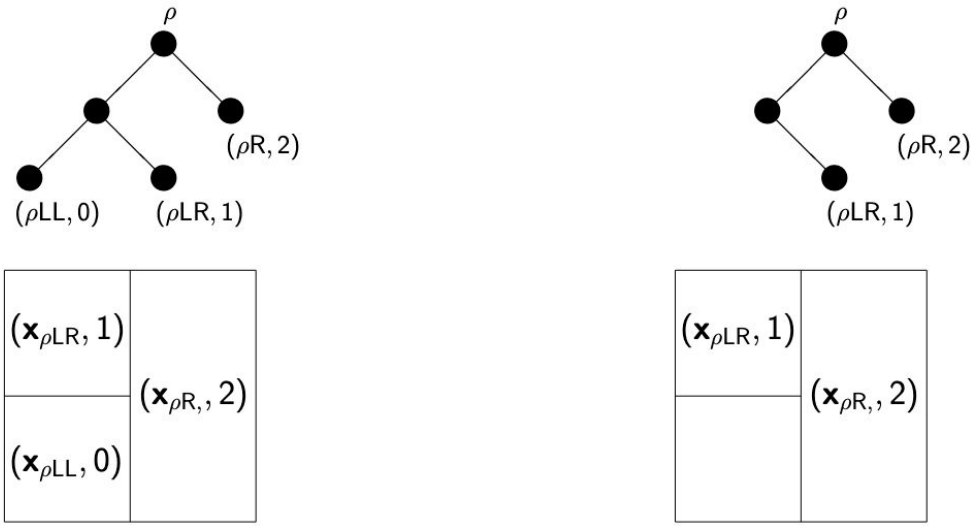


Figure: Sparse tree representation of $\mathbb{Z}$-MRP with identity element $e = 0$.

Arithmetic with Sparse Binary Trees (eg. Statistical regular pavings to analyze massive data of aircraft trajectories, Gloria Teng, Kenneth Kuhn and Raazesh Sainudiin, Journal of Aerospace Computing, Information, and Communication, Vol. 9, No. 1, pp. 14-25, doi: 10.2514/1.I010015, 2012).

SEE: http://lamastex.org/preprints/AAIASubPavingATC.ps

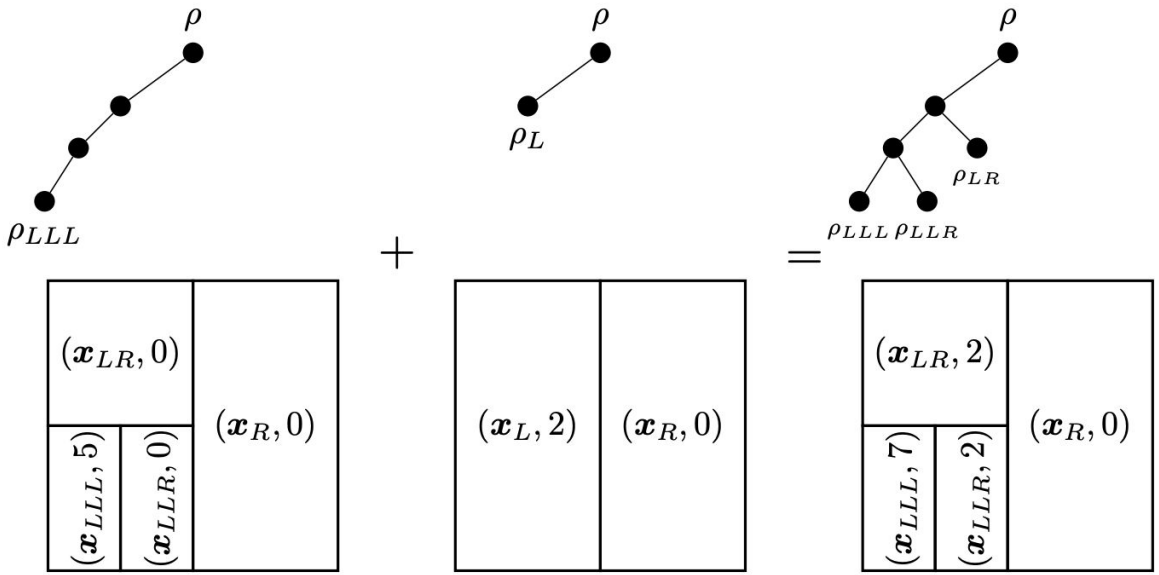


Figure 3.4: Operation on two sparse $\mathbb{Z}$-MRPs with operation $+$ and neutral element $e = 0$. Each box in the result gets augmented with the sum of the values of the boxes that overlaps with it in $s^{(1)}$ and $s^{(2)}$.

Statistically Equivalent Block Random PQ Markov chain

ALGORITHM 1: `SEBTreeMC`$(s, \overline{\#}, \overline{m})$

input : s, initial SRP with root node ρ,
$x = (x_1, x_2, \ldots, x_n)$, a data burst of size n,
$\underline{\#} : \mathbb{L}^{\triangledown}(s) \to \mathbb{R}$, a priority function of counts,
$\overline{\#}$, maximum value of $\#(\rho\mathsf{v}) \in \mathbb{L}^{\triangledown}(s)$ for any splittable leaf node in the final SRP,
$\overline{m}$, maximum number of leaves in the final SRP.

output : a sequence of SRP states $[s(0), s(1), \ldots, s(T)]$ such that $\mathbb{L}^{\triangledown}(s(T)) = \emptyset$ or $\#(\rho\mathsf{v}) \leq \overline{\#}\ \forall \rho\mathsf{v} \in \mathbb{L}^{\triangledown}(s(T))$ or $|\mathbb{L}(s(T))| = \overline{m}$.

initialize: $\boldsymbol{x}_\rho \rightsquigarrow x$, make $\boldsymbol{x}_\rho$ such that $\cup_i^n x_i \subset \boldsymbol{x}_\rho$ if $\nexists$ domain knowledge or historical data,
$s \rightsquigarrow \boldsymbol{x}_\rho$, specify the root box of s,
$\mathbf{s} \leftarrow [s]$

while $\mathbb{L}^{\triangledown}(s) \neq \emptyset$ & $|\mathbb{L}(s)| < \overline{m}$ & $\max_{\rho\mathsf{v} \in \mathbb{L}^{\triangledown}(s)} \#(\rho\mathsf{v}) > \overline{\#}$ **do**

$\rho\mathsf{v} \leftarrow$ `random_sample` $\left(\underset{\rho\mathsf{v} \in \mathbb{L}^{\triangledown}(s)}{\text{argmax}}\ \#(\rho\mathsf{v}) \right)$ `// sample uniformly from nodes with largest #`

$s \leftarrow s$ with node $\rho\mathsf{v}$ split `// split the sampled node and update` s

s.append(s) `// append the new SRP state with an additional split`

end

What do we provide and the sources of ideas and tools?

Software Tool
SparkDensityTree

Sandstedt, A., Graner, J., Wiklund, T., & Sainudiin, R. (2023).

SparkDensityTree: An Apache Spark Library for Scalable Density Estimation, Anomaly Detection and Conditional Density Regression with Universal Performance Guarantees Using Distributed Sparse Binary Trees (Version 1.0) [Computer software].

https://github.com/lamastex/SparkDensityTree
https://github.com/lamastex/SparkDensityTree-examples

SparkDensityTree's Mathematical Sources

- Built on the ideas of several white papers inspired by the following fields:
  - Constructive Mathematics
  - Set-Valued and Interval Analysis for Epistemologically Valid Machine Interval Experiments
  - Applied Interval Analysis for Autonomous Control Systems (eg. Luc Jaulin's Lab)
  - Distributed Algorithms and Optimization
- White Papers:
  - Mapped Regular Pavings
  - Posterior Expectation of Regularly Paved Random Histograms
  - An Auto-Validating, Trans-Dimensional, Universal Rejection Sampler for Locally Lipschitz Arithmetical Expressions
  - Minimum Distance Histograms with Universal Performance Guarantees (**UPG**)
  - Scalable Multivariate Histograms
  - Scalable Algorithms in Nonparametric Computational Statistics,
    - Johannes Graner's MSc thesis
  - Scalable Nonparametric L1 Density Estimation via Sparse Subtree Partitioning,
    - Axel Sandstedt's MSc thesis (to be released)

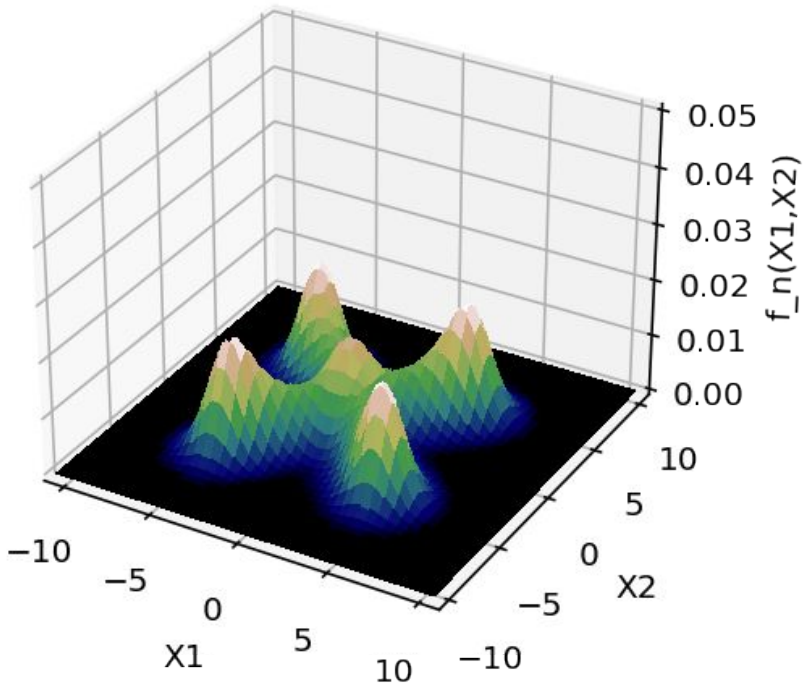
What is it good for?

SparkDensityTree – A Software Tool

- Constructs density estimates in a distributed manner
- Provide a Sparse Tree based Histogram with UPG
- To provide the user with several tools for use in
  - Anomaly detection via
    - generalised tail probabilities and
    - highest density level-sets
  - Conditional density regression
    - Predictive sampling from training data
  - Arithmetic over an algebra of sparse trees that is dense in C_0
    - by Stone-Weierstrass Theorem

Estimate of a Mixture of Gaussians density (2D Cross)

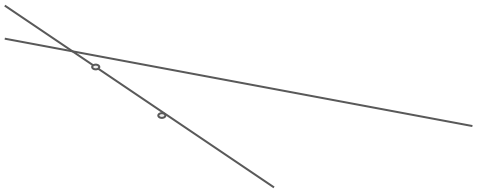
Using 100GB of samples from a Gaussian Mixture on a Cross of two lines in 2D, We first construct an optimally smoothed joint density estimate as a histogram from the space of sparse binary trees, as shown:

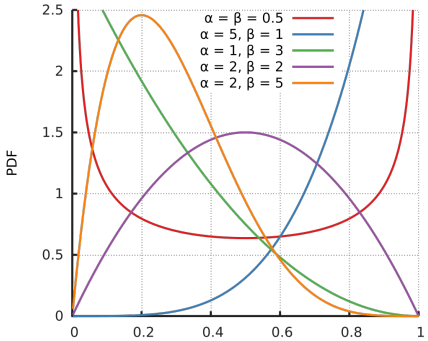


Beta-Weighted LineMixtures of Scaled Gaussian locations

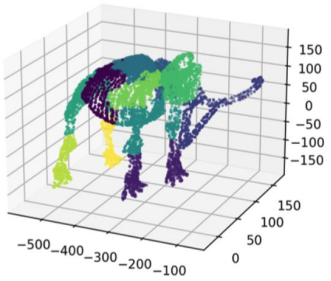
```
case class LineMixture(start : Array[Double], end : Array[Double], numMixtures : Int,
                       alpha : Double, beta : Double, scales : Array[Double])

@param lineWeights - positive weights corresponding to non-normalized probabilities
case class MixtureDistribution(lineMixtures : Array[LineMixture], lineWeights : Array[Double])
```

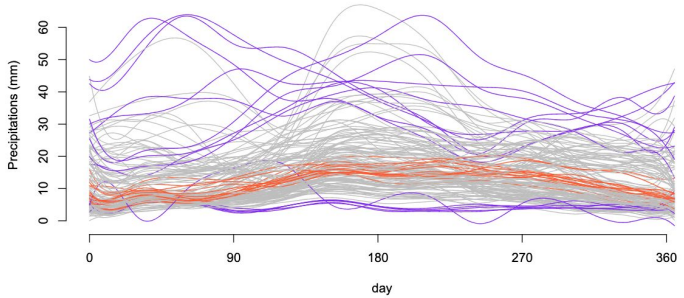




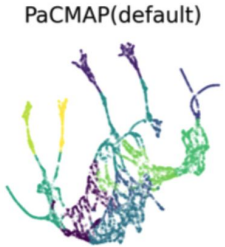
Dimensionality Reduction Methods allow projections even from infinite dimensional functional data

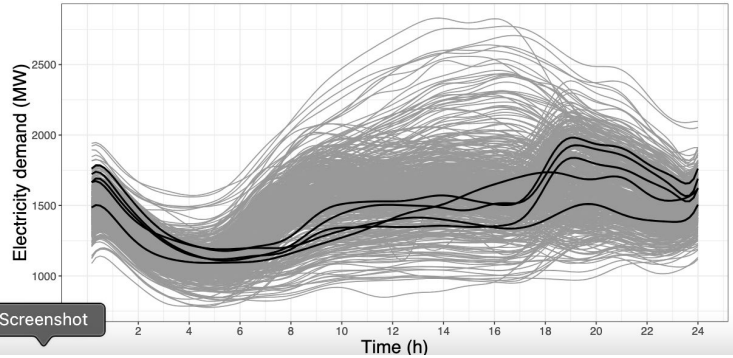


(a)



(b)





Screenshot

Regression Analysis

- What is regression, care? [Analyze $f_n(X_2|X_1)$ with dimension d=2]
- How to do regression using the joint density estimate $f_n(X_1,X_2)$?
- We can check for anomalies, points found in low probability regions
- First, we may check how any specific set of variables are concen.
- We can predict outcomes of needed variables C given a set of fixed variables D, where C and D are mutually exclusive subsets of all observed variables in [d] := {1,2,..,d} of the training and validation datasets.
- Note that $E(X_C \mid X_D)$ is not good enough for prediction, instead we can sample from actual estimate of the conditional density $f_n(X_C \mid X_D)$ from the optimally smoothed joint density estimate $f_n(X_1,X_2,...,X_n)$

Regression Analysis

- Thus we can analyze the relationship between variables and their distribution
- Can help us in finding anomalies
- Generate new reasonable data from underlying distribution
- A Real Dataset FX-1 Minute Data over many decades! (but we simulate data here)

For Which Pairs?

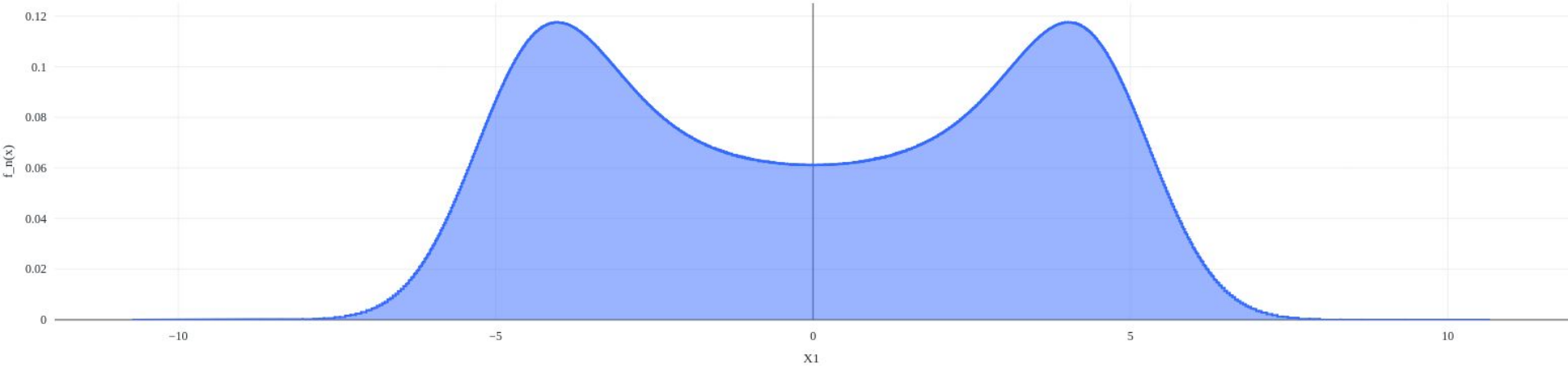
Currently we have available historical data for the following Forex Pairs:

EUR/USD, EUR/CHF, EUR/GBP, EUR/JPY, EUR/AUD, USD/CAD, USD/CHF, USD/JPY, USD/MXN, GBP/CHF, GBP/JPY, GBP/USD, AUD/JPY, AUD/USD, CHF/JPY, NZD/JPY, NZD/USD, XAU/USD, EUR/CAD, AUD/CAD, CAD/JPY, EUR/NZD, GRX/EUR, NZD/CAD, SGD/JPY, USD/HKD, USD/NOK, USD/TRY, XAU/AUD, AUD/CHF, AUX/AUD, EUR/HUF, EUR/PLN, FRX/EUR, HKX/HKD, NZD/CHF, SPX/USD, USD/HUF, USD/PLN, USD/ZAR, XAU/CHF, ZAR/JPY, BCO/USD, ETX/EUR, EUR/CZK, EUR/SEK, GBP/AUD, GBP/NZD, JPX/JPY, UDX/USD, USD/CZK, USD/SEK, WTI/USD, XAU/EUR, AUD/NZD, CAD/CHF, EUR/DKK, EUR/NOK, EUR/TRY, GBP/CAD, NSX/USD, UKX/GBP, USD/DKK, USD/SGD, XAG/USD, XAU/GBP

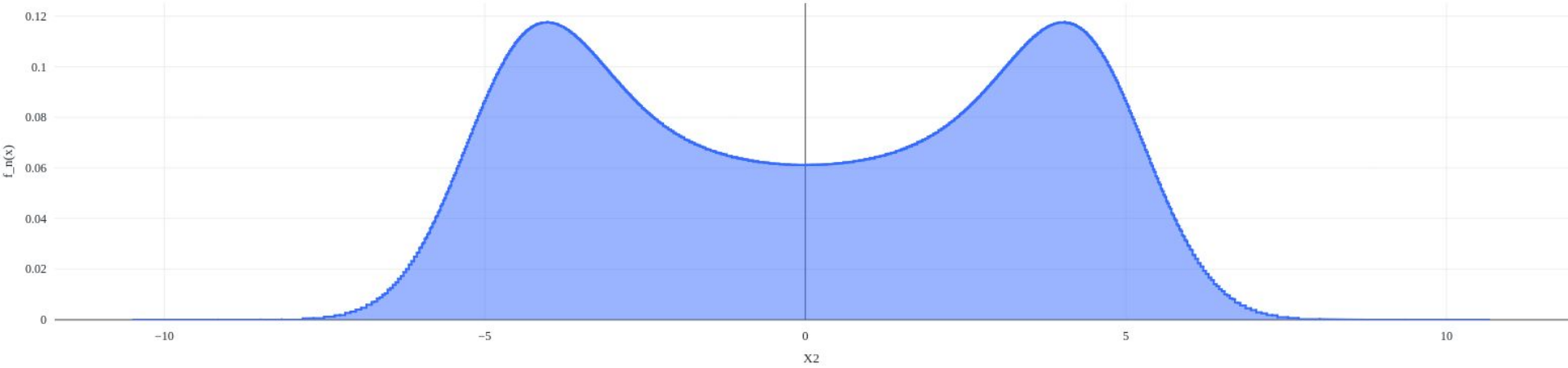
Marginal Densities from Integrating Joint Density

- See how a certain set of variables are concentrated along a subset of {1,...,d}.
- Let us see examples of the marginal X_1 and marginal X_2 for the 2D-Cross

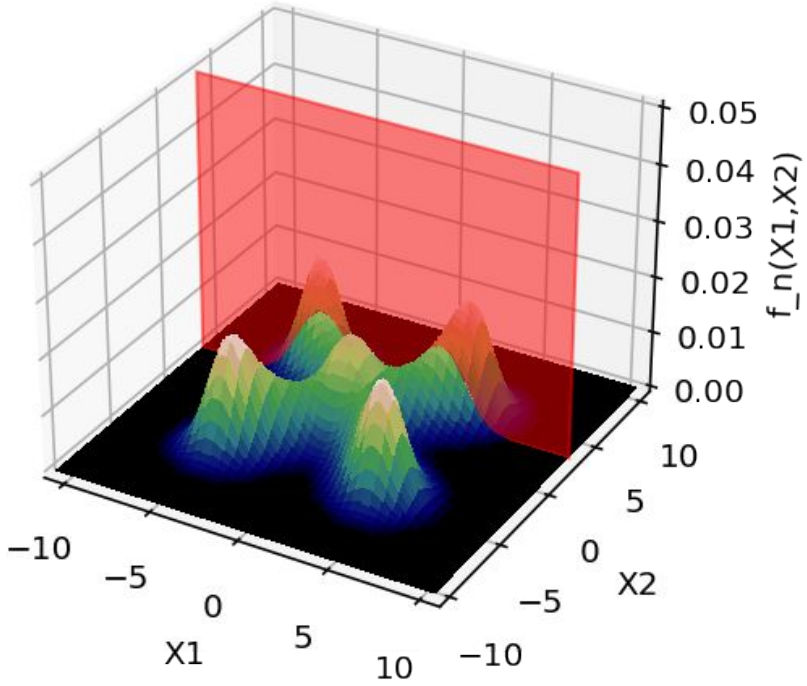
Marginal density of x_1 from joint density $f_n(X_1,X_2)$ (via sparse trees)

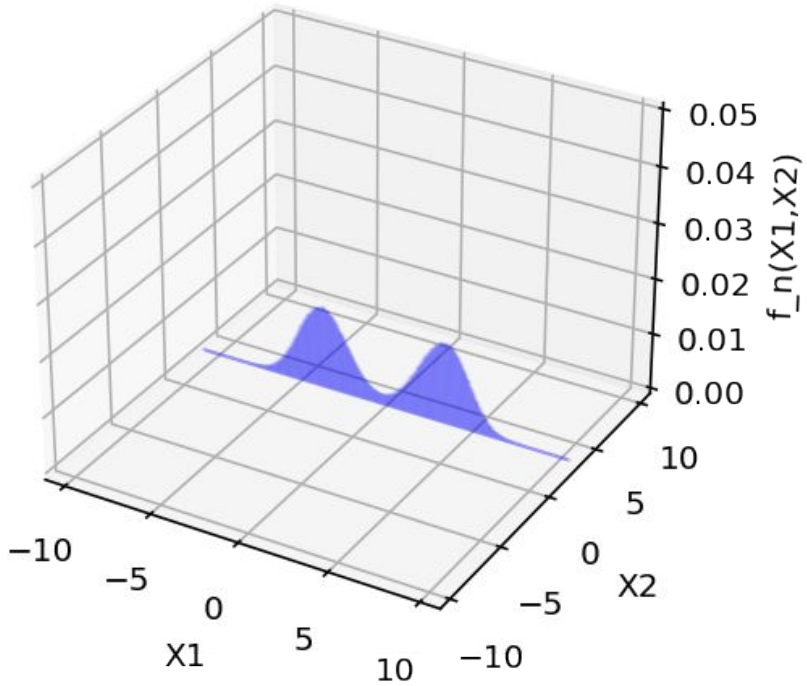


Marginal density of x2 from joint density f (via sparse tree ops)



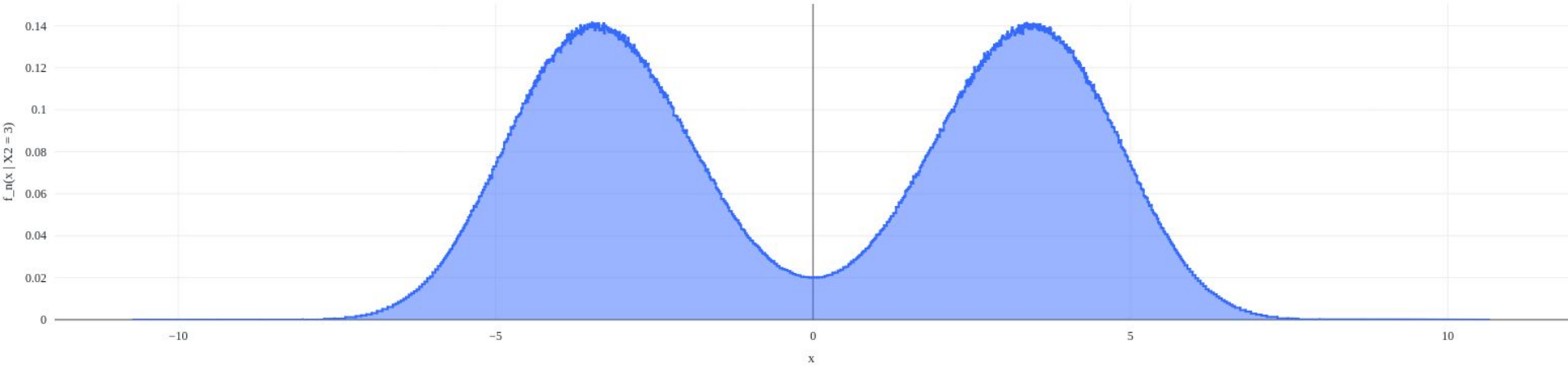
Conditional Densities



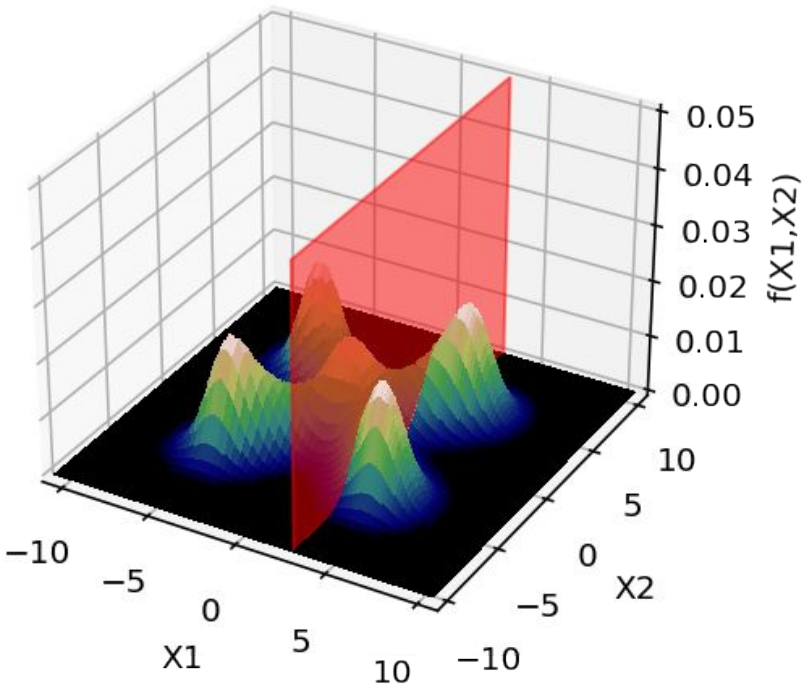


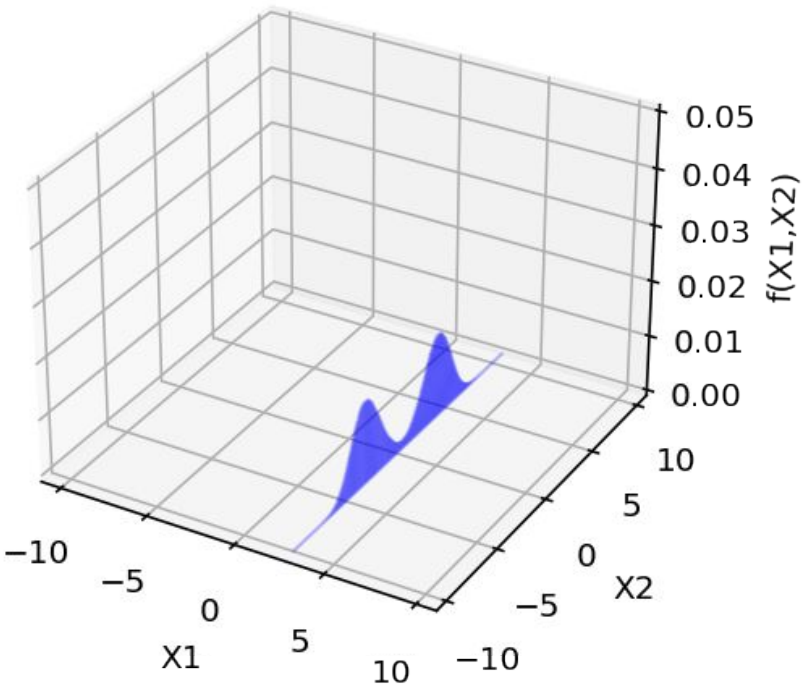
Conditional Densities

- Normalizing the slice gives us a new probability distribution at the fixed point

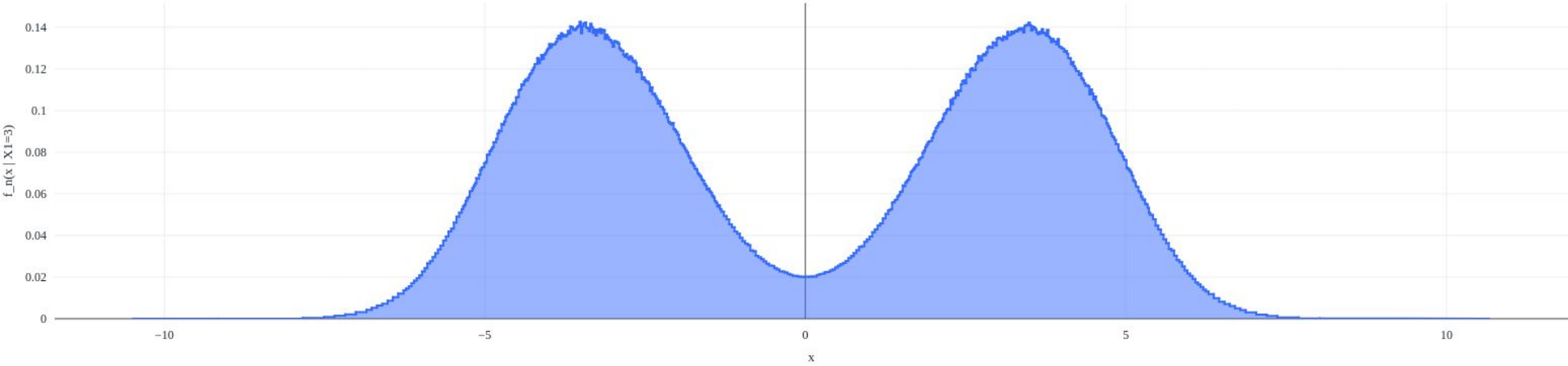


Conditional Density Estimate from Joint Density Estimate





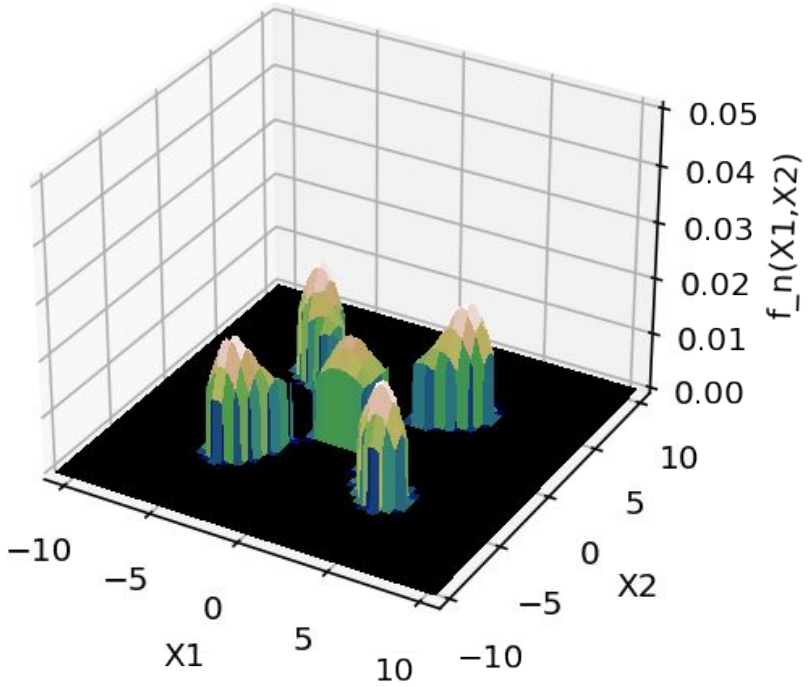
Conditional Densities

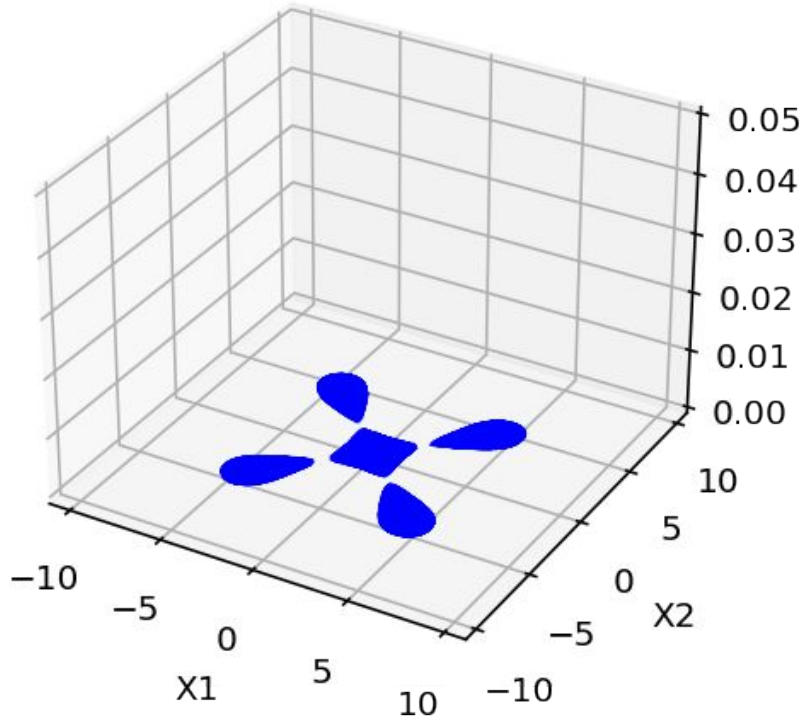


Conditional Densities

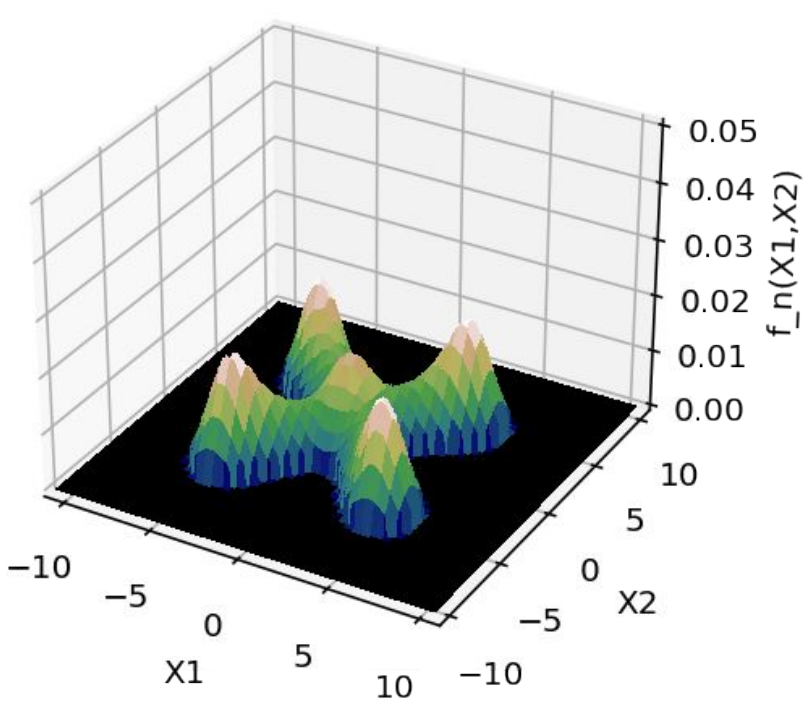
- Can be done for any dimension, any set of fixed axes

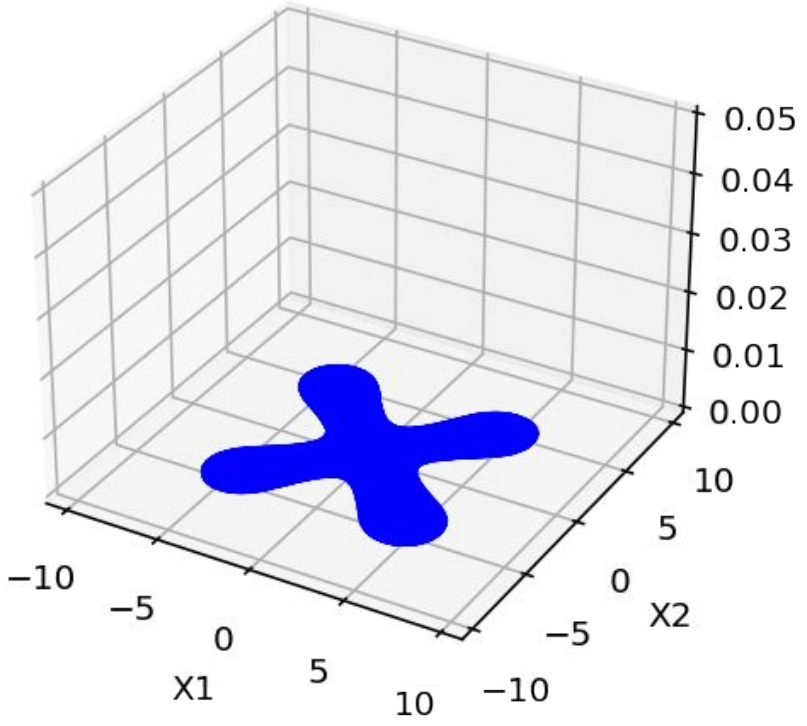
50%-highest density region



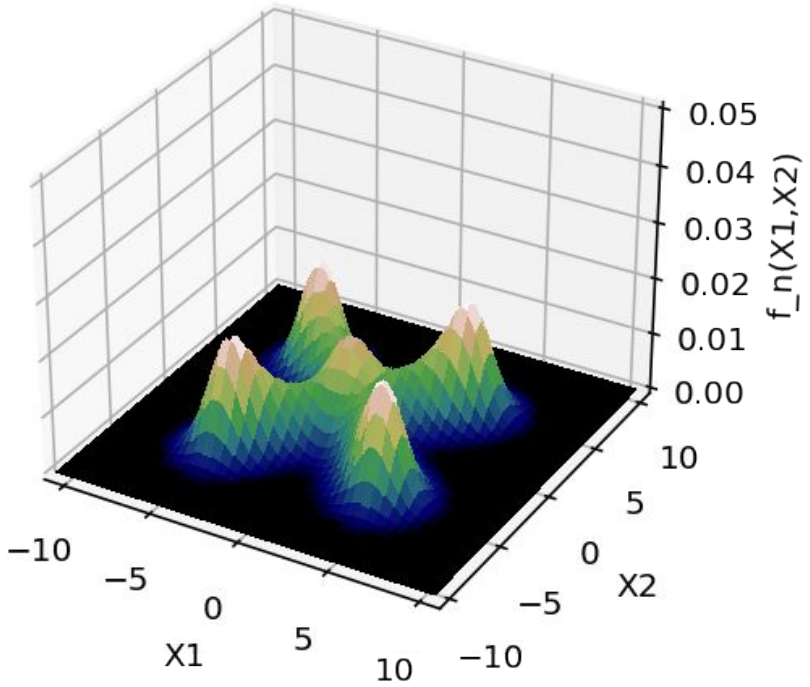


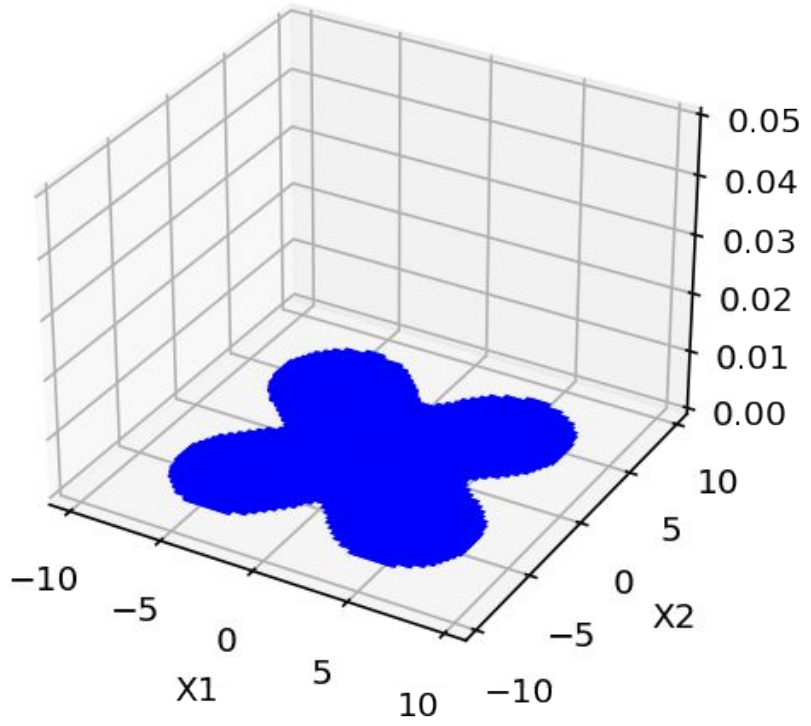
90%-highest density region





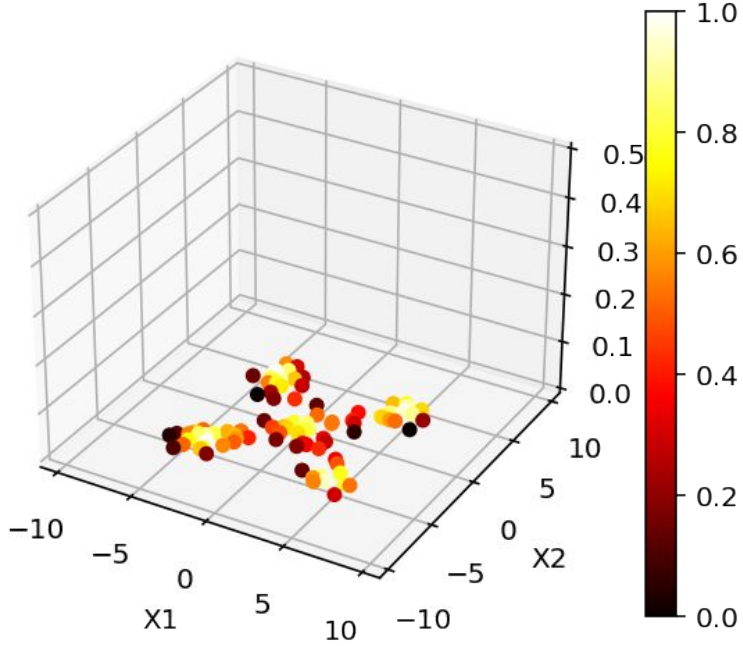
99.9%-highest density region



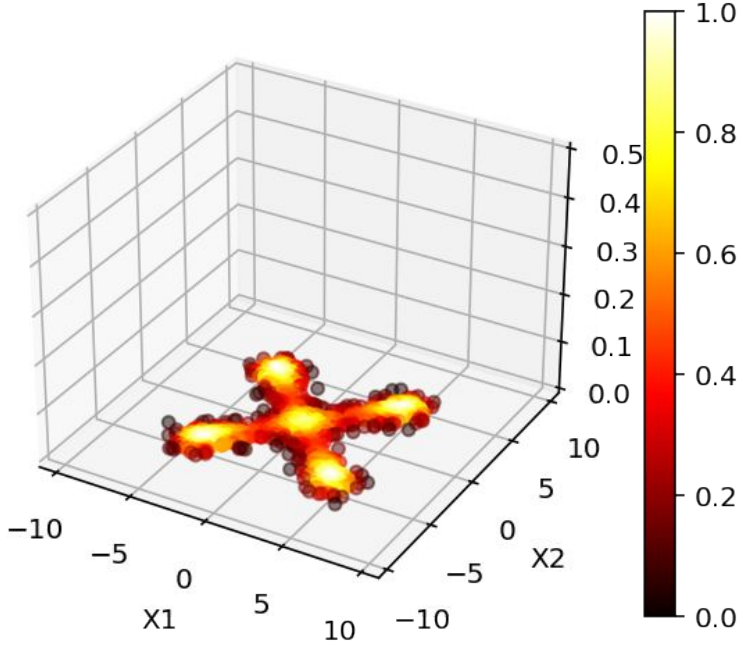


Anomaly Detection

100 data points sampled

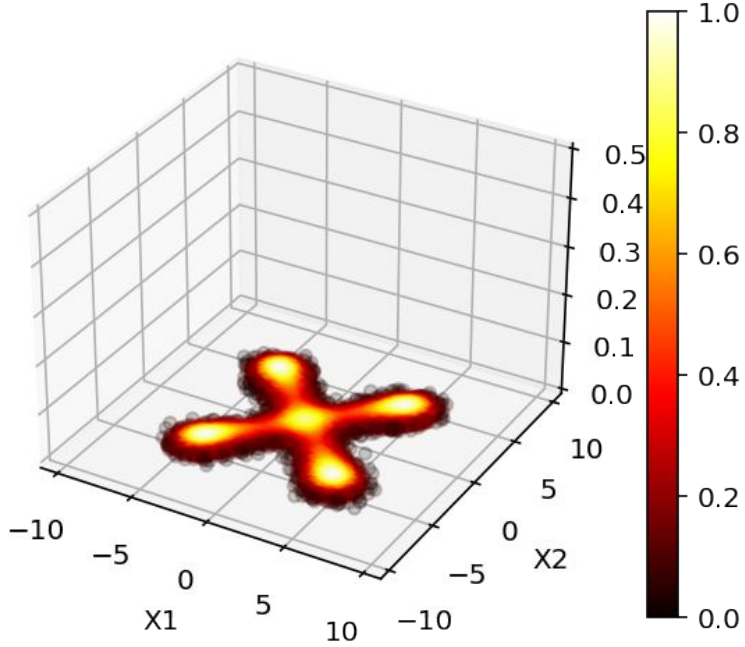


1000 data points sampled

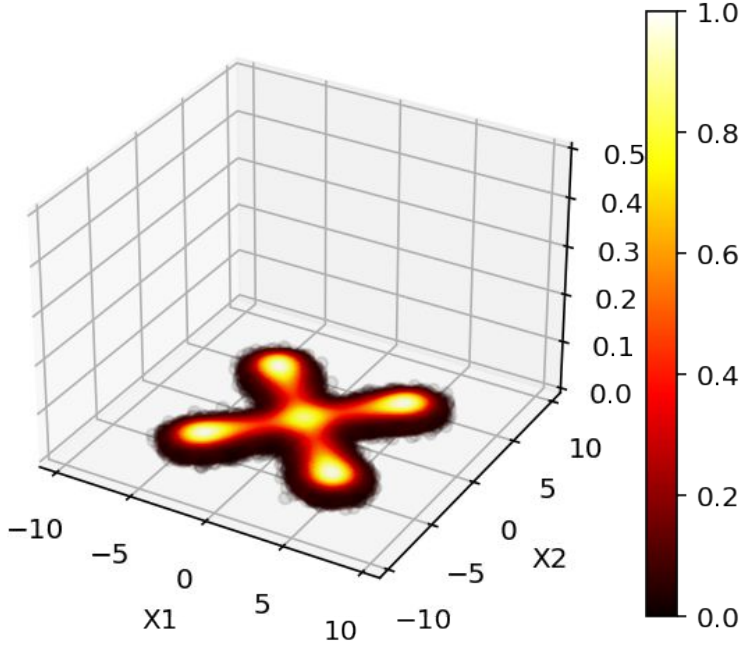


Anomaly Detection

10000 data points sampled



100000 data points sampled



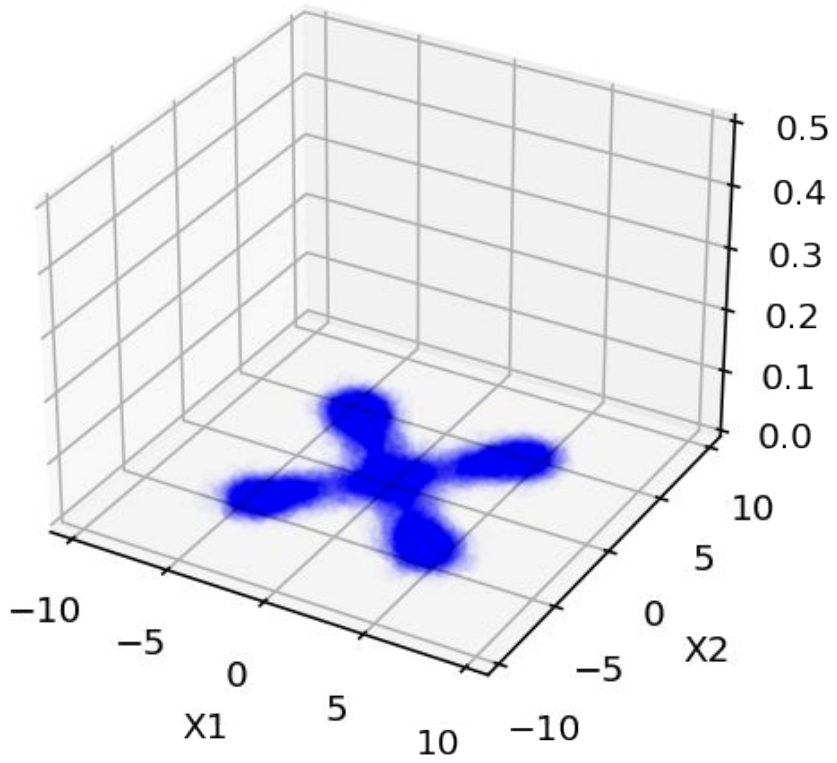
Highest density level-sets / Tail Prob for Anomaly Detection

- We generate a sample of 10^6 points to see how what proportion of points fall into each region:

Wanted Probability	Actual Probability	Proportion within region
0.90	0.900002	0.900065
0.95	0.950002	0.949944
0.99	0.990002	0.989812
0.999	0.999003	0.998907

Sampling from the estimator of the 2D Cross density

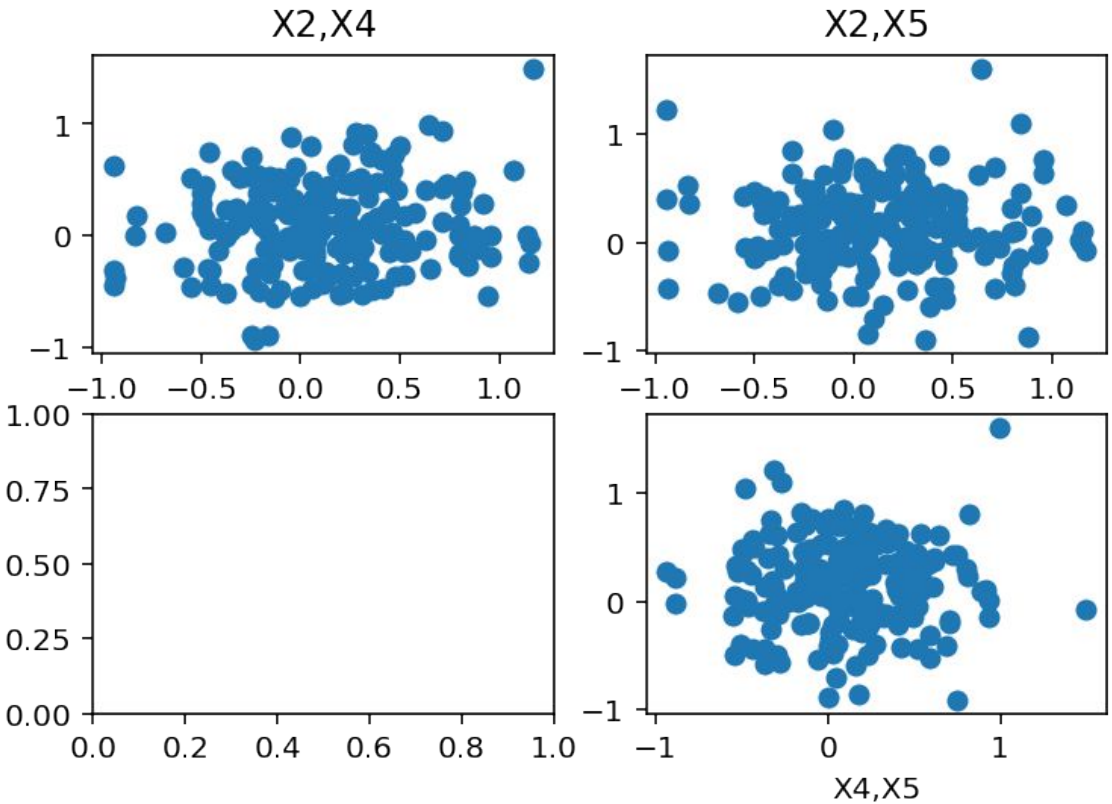
- 10000 data points



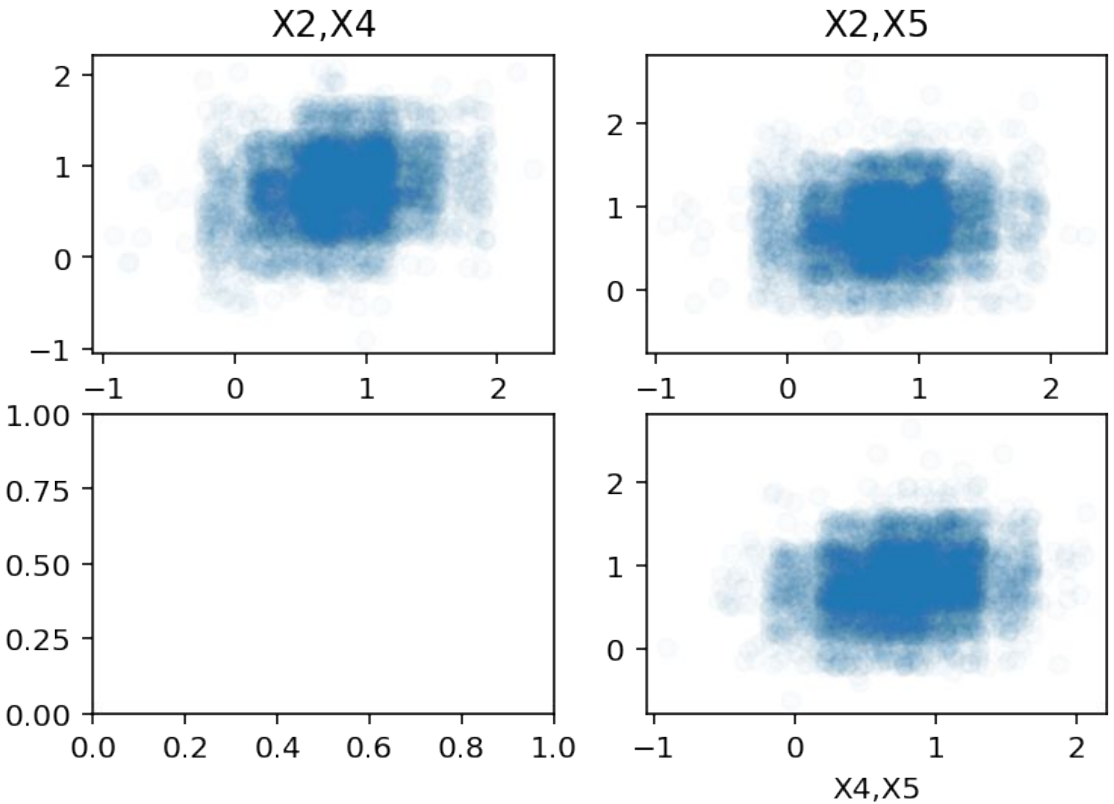
10 dimensional mixture conditional sampling

- 2-line 10 dimensional cross:
- Line 1: (0, …, 0) -> (1, …, 1)
- Line 2: (1, 0, 1, 0, ... , 0) -> (0, 1, 0, 1, …, 1)

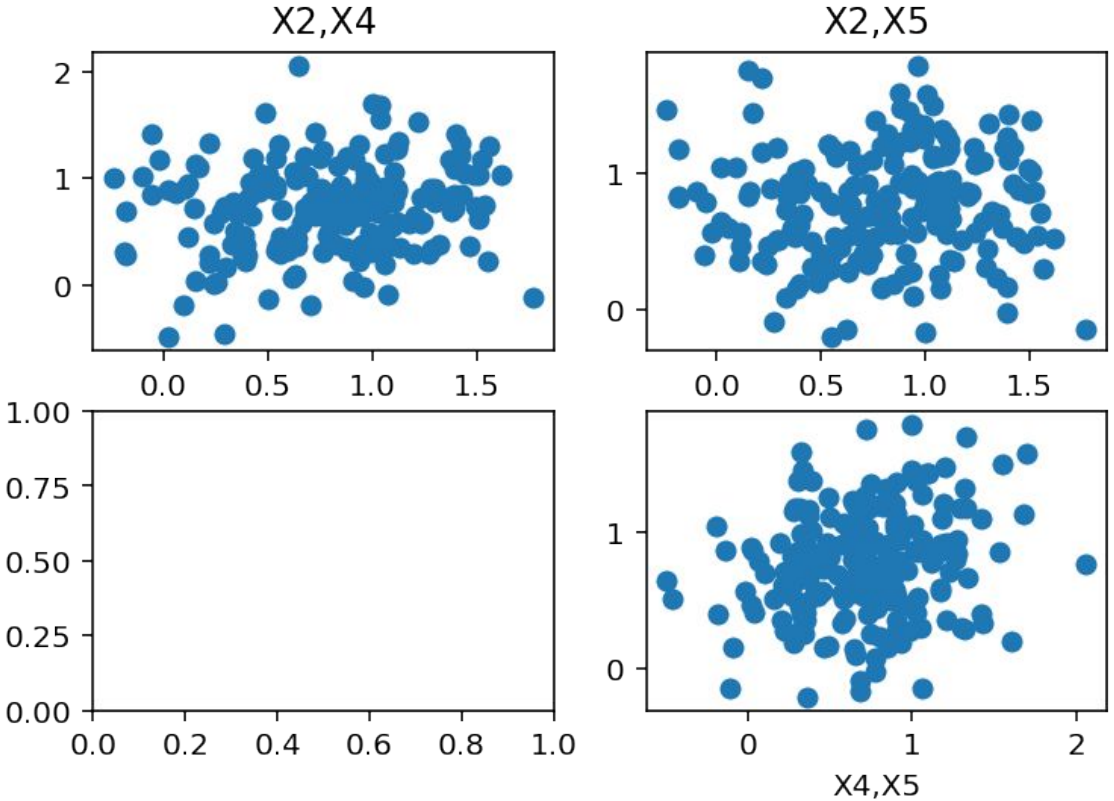
$$f_n(X_2, X_4, X_5 \mid X_1 = 0, X_3 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0, X_{10} = 0)$$



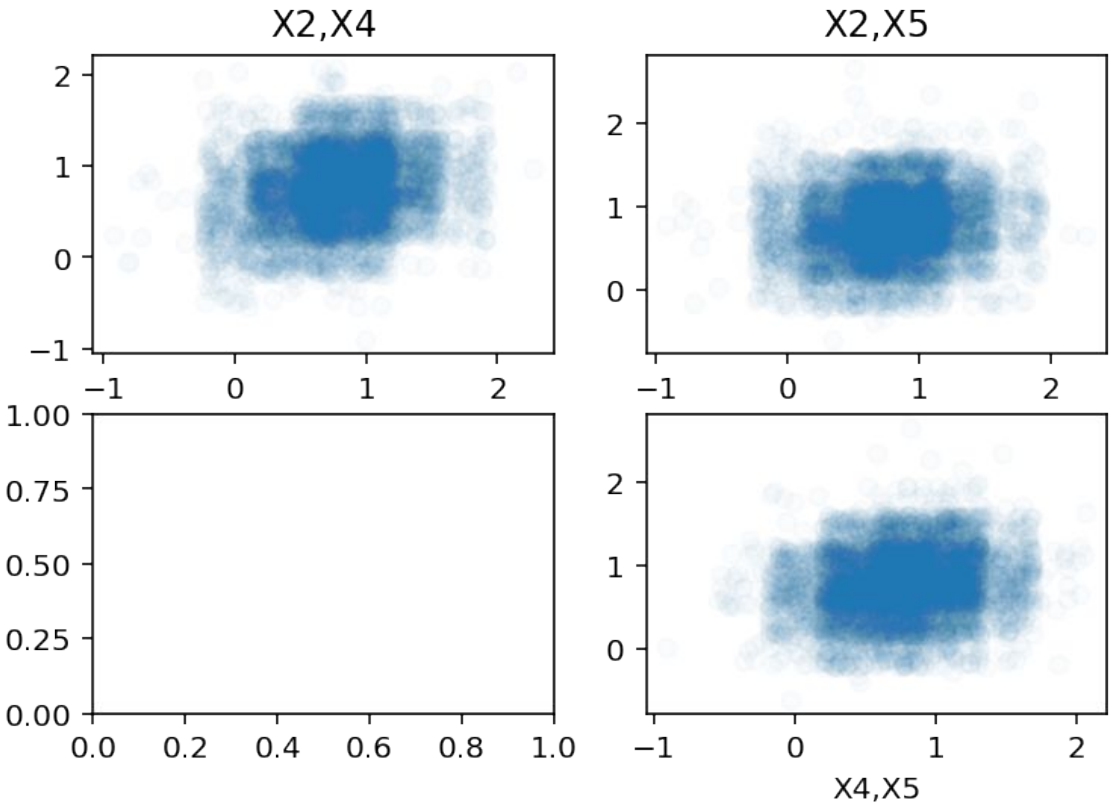
$$f_n(X_2, X_4, X_5 \mid X_1 = 0, X_3 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0, X_{10} = 0)$$



$$f_n(X_2, X_4, X_5 \mid X_1 = 0.825, X_3 = 0.825, X_6 = 0.825, X_7 = 0.825, X_8 = 0.825, X_9 = 0.825, X_{10} = 0.825)$$



$$f_n(X_2, X_4, X_5 \mid X_1 = 0.825, X_3 = 0.825, X_6 = 0.825, X_7 = 0.825, X_8 = 0.825, X_9 = 0.825, X_{10} = 0.825)$$



Results

- 256 cores, 480GB memory (16 c4.4xlarge workers)
- 70/30 split between training data and hold-out data
- For 2 dimensions, n = 62,500,000,000
- For 10 dimensions, n = 12,500,000,000

Distribution	Dimensions	sample size	finestResDepth	maxLeafCount	φ	total time (h:m)
Uniform	2	1TB	60	2	100000	07:11
Uniform	10	1TB	60	6	100000	00:45
Cross Mixture	2	1TB	44	3	100000	04:42
Cross Mixture	10	1TB	60	5	20000	00:46

Scalability

- We have linear scaling with doubling of worker cores
- Again, 70/30 split between training data and hold-out data
- For 2 dimensions, n = 6,250,000,000
- For 10 dimensions, n = 1,250,000,000

Distribution	Dimensions	sample size	num cores	countLimit	total time (h:m:s)
Uniform	2	100GB	64	10000	01:30:08
Uniform	2	100GB	128	10000	00:49:34
Uniform	2	100GB	256	10000	00:28:23
Cross Mixture	2	100GB	64	10000	01:43:57
Cross Mixture	2	100GB	128	10000	01:00:13
Cross Mixture	2	100GB	256	10000	00:37:26

Breakdown of 1TB timings

Distribution	dim	Rect partitions	labelTrain partitions	maxLabel partitions	mergeRDD partitions	labelValid+getMDE partitions
Uniform	2	00:06:47 4096	04:08:00 4096	00:03:25 4096	02:53:00 16384	00:16:05 4096
Uniform	10	00:06:34 8192	00:15:40 8192	00:00:45 8192	00:19:39 8192	00:02:41 4096
Cross Mixture	2	00:07:13 16384	01:35:00 16384	00:03:01 16384	02:22:00 16384	00:34:30 4096
Cross Mixture	10	00:04:22 8192	00:13:59 8192	00:00:43 8192	00:21:03 8192	00:06:10* 4096

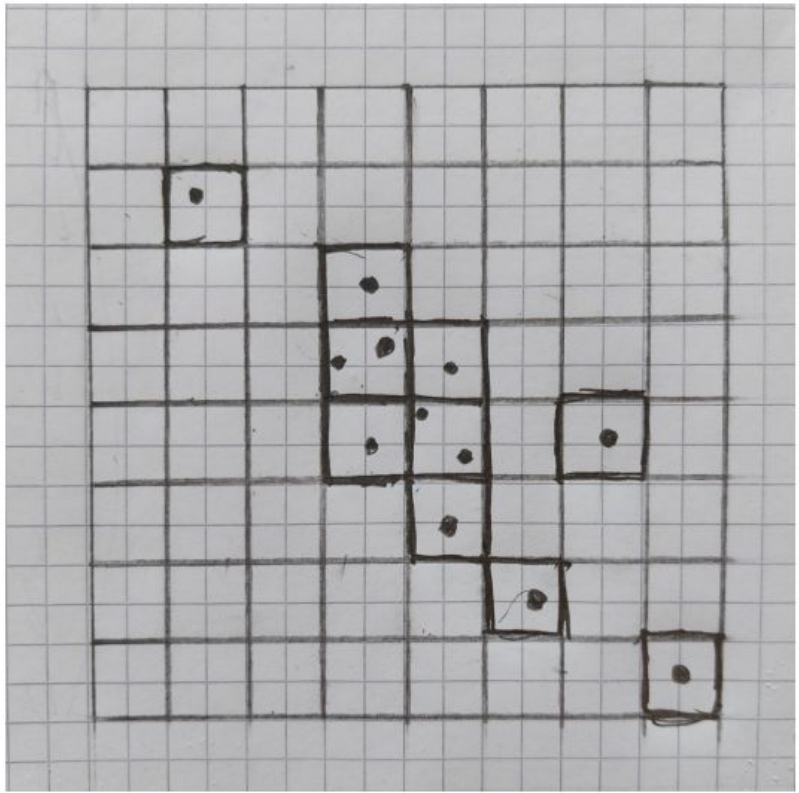
How do we do it?

Construction of an estimate

- Stage 1: Obtain Box hull that contains all the datapoints dimension-wise:
  - $[[\min \{x_{i,k}\}_{i=1,\ldots,n}, \max\{x_{i,k}\}_{i=1,\ldots,n}]: k=1,\ldots,d\}]$, a pure map-reduce step
- Stage 2: Leaf Address/Label: Map each data point to its binary leaf address
  - This is the (precision-loss-able) data compression stage: *finestResDepth* and *maxLeafCount*
- Stage 3: Merging to finest histogram considerable due to driver RAM budget
  - This is specified by *countLimit*, giving the maximum number of datapoints in any leaf of the coarsest histogram tree to be further coarsened in driver to find the optimal MDE histogram
- Stage 4: Search for MDE on a Markov chain's path of coarsening histograms
  - Using collator regular pavings that are broadcasted over workers for distributed validation
  - Universal Performance guarantees are given by the SEB-PQ-Markov chain and the Vapnik-Chervonenkis dimension of the Yatracos Class of Scheffe sets of the sparse tree histogram along the coarsening path of the chain.

- Note that splitting can be distributedly, split everything in any order down to $\overline{\#}$
- Sparse representation allows us to split extremely refined, number of leaves bounded by number of points

- Bottom-up approach
- Split everything down to some very large depth
- Backtrack to a valid output of SEBTreeMC by merging cells up to the count limit



The whole method can be described in four seperate stages:

- Retrieve Hull of $X_1, \ldots, X_n$
- Find leaf labels of $X_1, \ldots, X_n$ at depth d, get count of each label
- Backtrack from depth d to valid output from SEBTreeMC
- Find the best performing estimate along the path

We focus on optimising distributed backtracking.

- Abstracts a lot of distributed computing details
- Machines are assigned *partitions* of data
- One machine is the driver, assign tasks to workers and so on.
- Other machines are workers which perform tasks and communicates to other machines in strict ways
- We can control what data gets assigned to what partition using *Partitioners*
- Less communication needed is good

- Issue: Previous backtracking does not account for data layout; we might need to merge two points on different machines; leads more communication between machines
- Solution: Find large subtrees of data and assign them to

Rough outline is as follows:

- (1) Sample data from partitions, each leaf l_i is assigned weight w_i
- (2) Sort the data accoring to the left-right ordering
- (3) Find large subtrees s_j according to a weight limit
- (4) Map all s_j to a new of partitions $p_1, \ldots, p_m$
- (5) map each leaf l_i to the same partition as the closest subtree s_j generated

- Machine 1 (partition 1) contain odd leaves
- Machine 2 (partition 2) contain even leaves

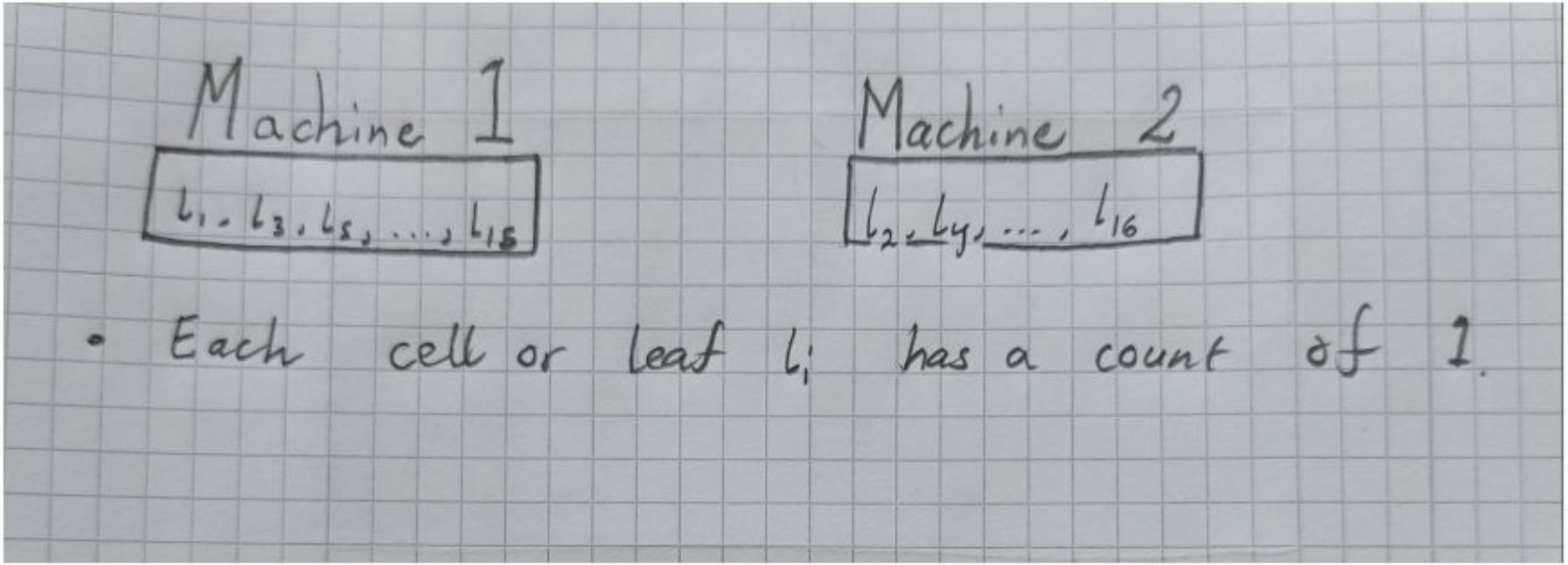


Figure: Layout of data partitions between machine 1 and 2.

- Leaf distribution at depth 4.

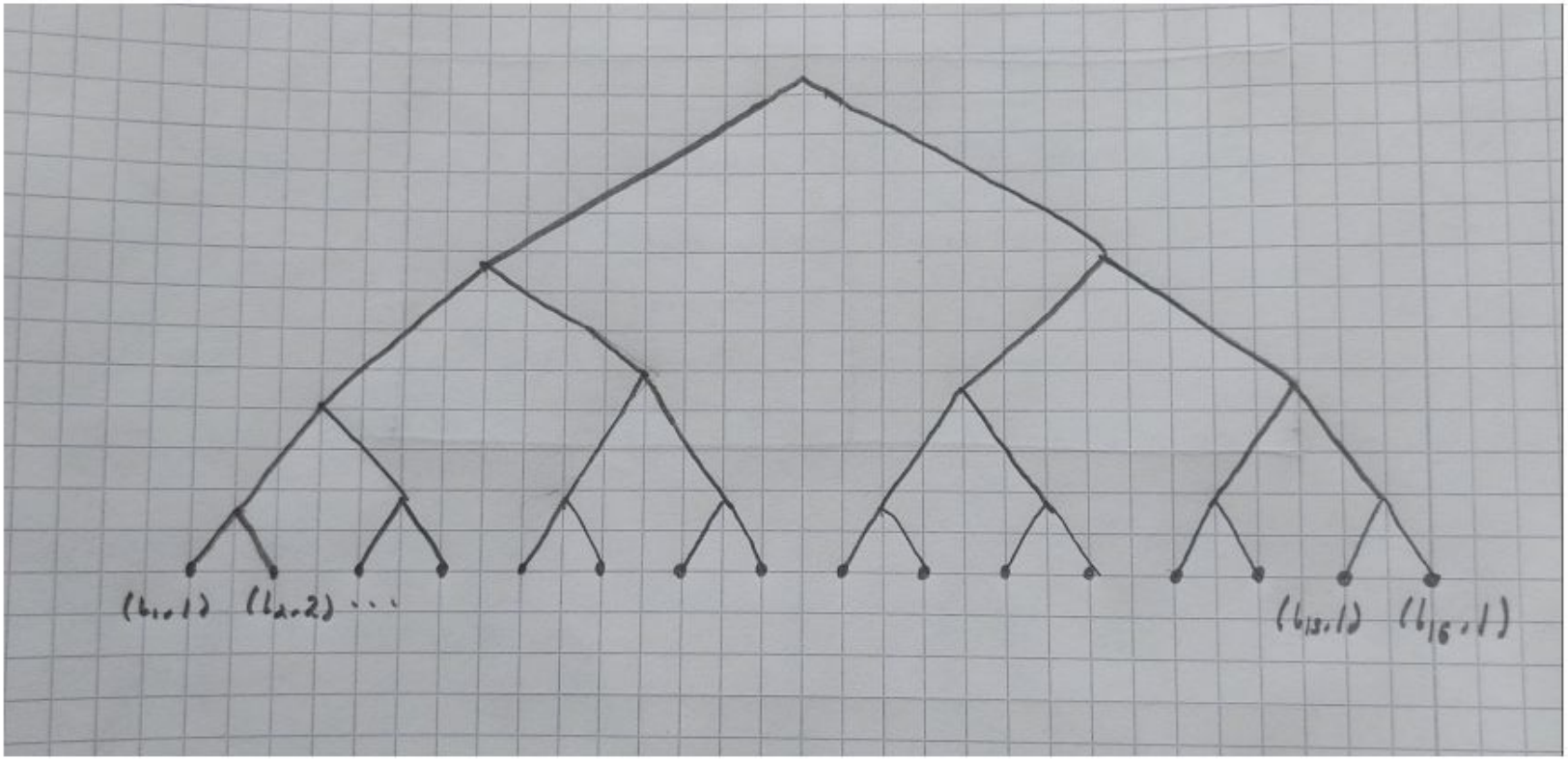


Figure: Tree layout of data between machines.

- Sample $l_1, l_3, l_9, l_{11}, l_{13}$ from partition 1 $\rightarrow$ weight $= \frac{8}{5}$.
- Sample l_2, l_4, l_{14} from partition 2 $\rightarrow$ weight $= \frac{8}{3}$

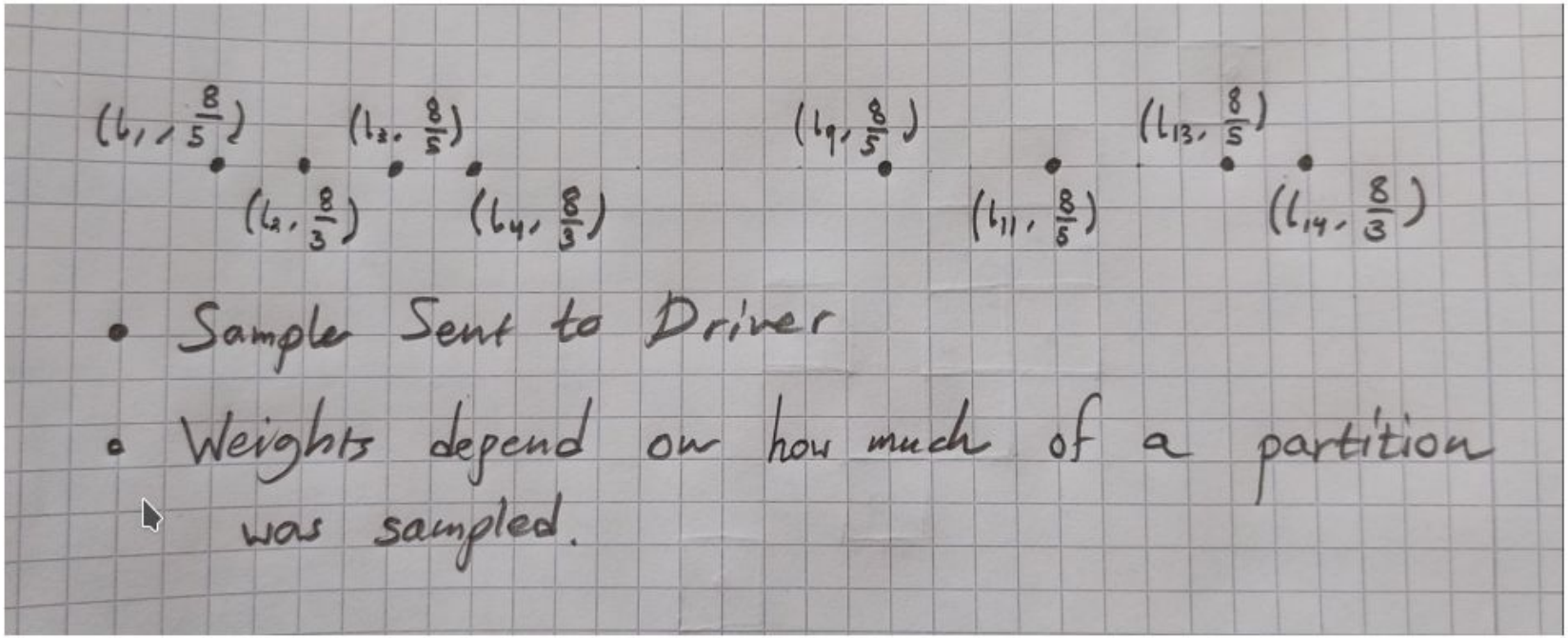


Figure: Leaf samples from the two machines..

- Find large nonintersecting subtrees with respect to weight from sample
- Same idea as backtracking
- O(nlog(n)) of input
- record the deepest subtree

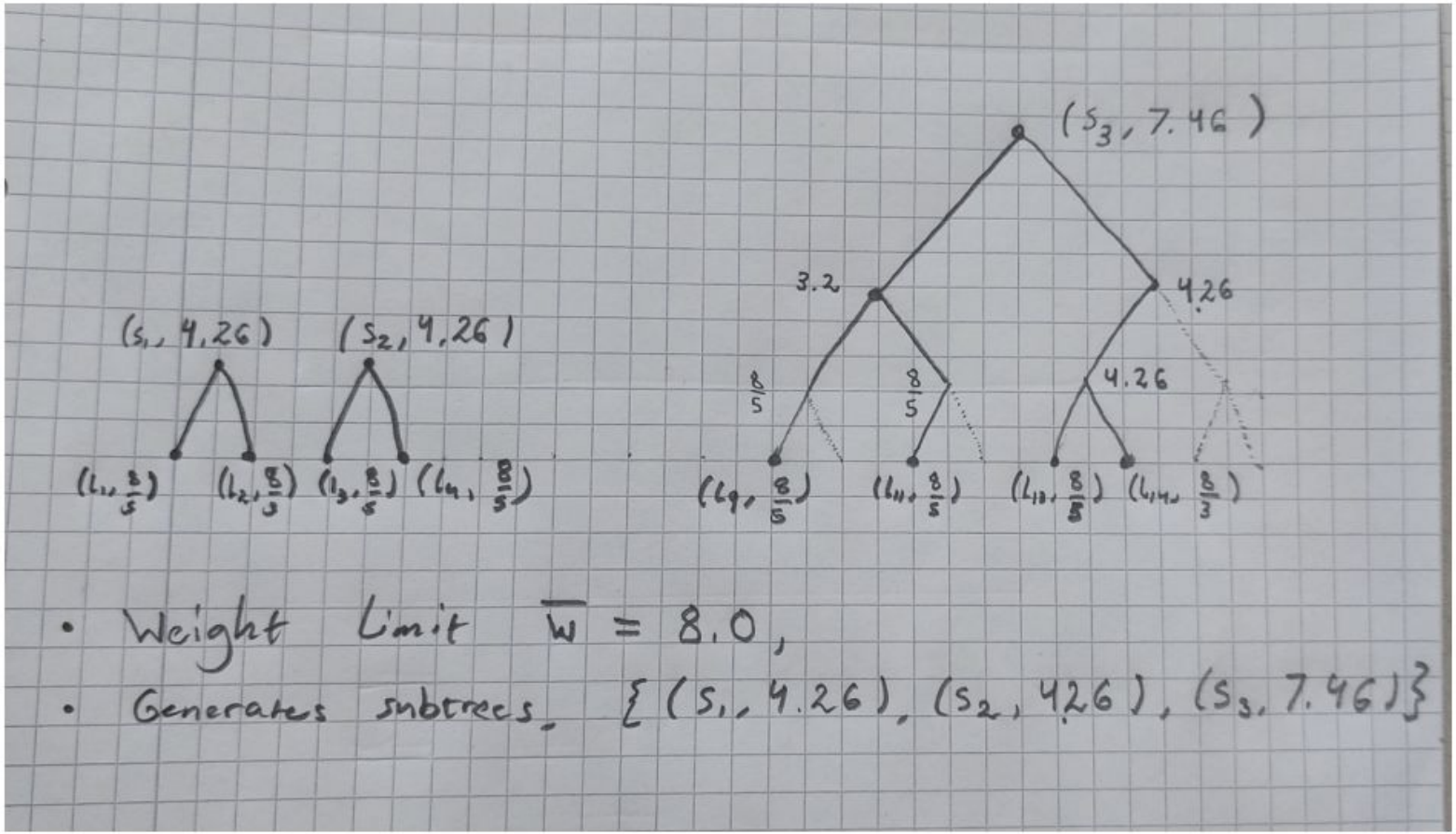


Figure: Generated maximal subtrees using weight limit $\overline{w} = 8.0$.

Map each subtree to some machine is some way to distribute the weight among machines:

Trees Left	Partition 1	Partition 2
$(s_1, 7.46), (s_2, 4.26), (s_3, 4.26)$	$\emptyset$	$\emptyset$
$(s_2, 4.26), (s_3, 4.26)$	$(s_1, 7.46)$	$\emptyset$
$(s_3, 4.26)$	$(s_1, 7.46)$	$(s_2, 4.26)$
$\emptyset$	$(s_1, 7.46)$	$(s_2, 4.26), (s_3, 4.26)$

Table: Assignment of generated subtrees to partitions.

- (1) Find the two closest subtrees of leaf l_i to the left and the right.
- (2) l_i is mapped to same partition the deepest subtree subtree is mapped to.
- (3) When leaf map is done, 1 all-to-all communication to send all leaves to correct machines.

Partition 1	$\{l_9, \ldots, l_{16}\}$
Partition 2	$\{l_1, \ldots, l_8\}$

Table: Assignment of leaves to partitions.

- Now all machines can locally do its backtracking up to largest recorded depth among trees
- When all merging is done locally, if any more merging need to take place, signal driver
- Do any last merges on the driver

Pros

- Constructed specifically for the distributed setting in mind
- Works for any continuous probability distribution (f in L)
- Works for any number of dimensions
- Very nice regression tools once you have constructed an estimate

Cons

- Hard to generate a density estimate (f_n of f), many parameters for the user to tune
- The construction can be seen as a four-stage process which makes the window for errors large

Some ideas to further improve performance in computational efficiency

- 70% in garbage collection of JVM process
- Modified algorithmic improvements

Thanks for your attention!