Scalable Kernel Density Classification via Threshold-Based Pruning

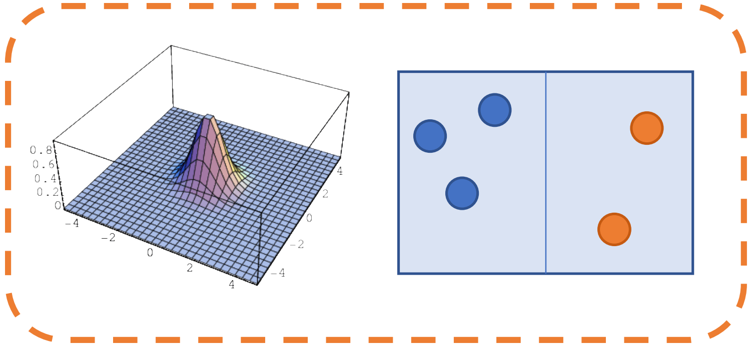
Edward Gan & Peter Bailis

MacroBase: Analytics on Fast Streams

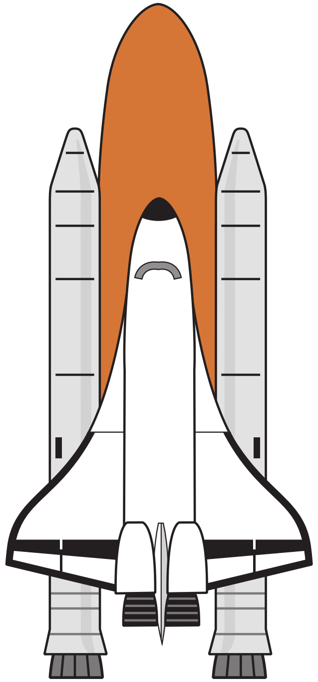
- Increasing Streaming Data
  - Manufacturing, Sensors, Mobile
  - Multi-dimensional + Latent anomalies

- Running in production
  - see CIDR17, SIGMOD17

- End-to-end operator cascades for:
  - Feature Transformation
  - Statistical Classification
  - Data Summarization

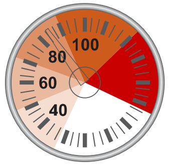


Example: Space Shuttle Sensors



[UCI Repository]

8 Sensors Total



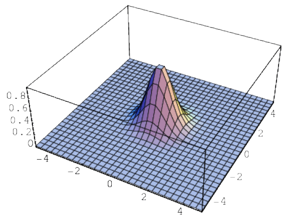
"Fuel Flow"

"Flight Speed"

Speed	Flow	Status
28	27	Fpv Close
34	43	High
52	30	Rad Flow
28	40	Rad Flow
…	…	

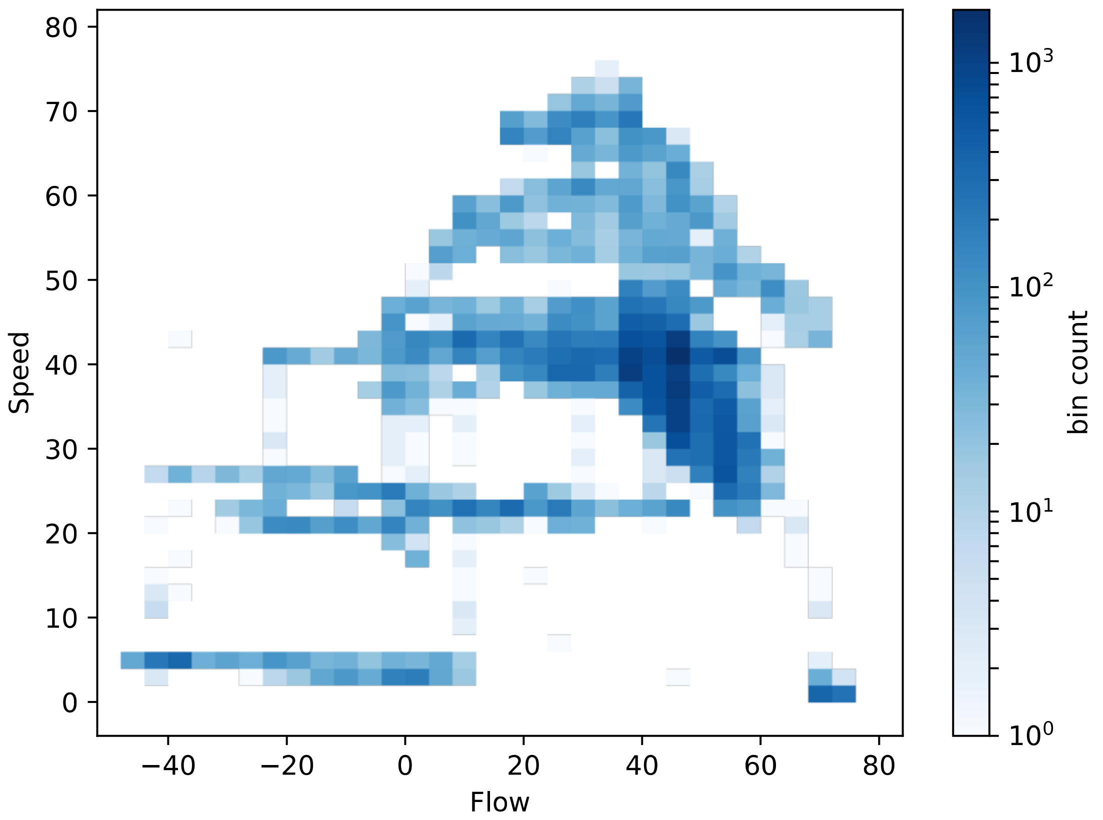
End-Goal: Explain anomalous speed / flow measurements.

Problem: Model distribution of speed / flow measurements.

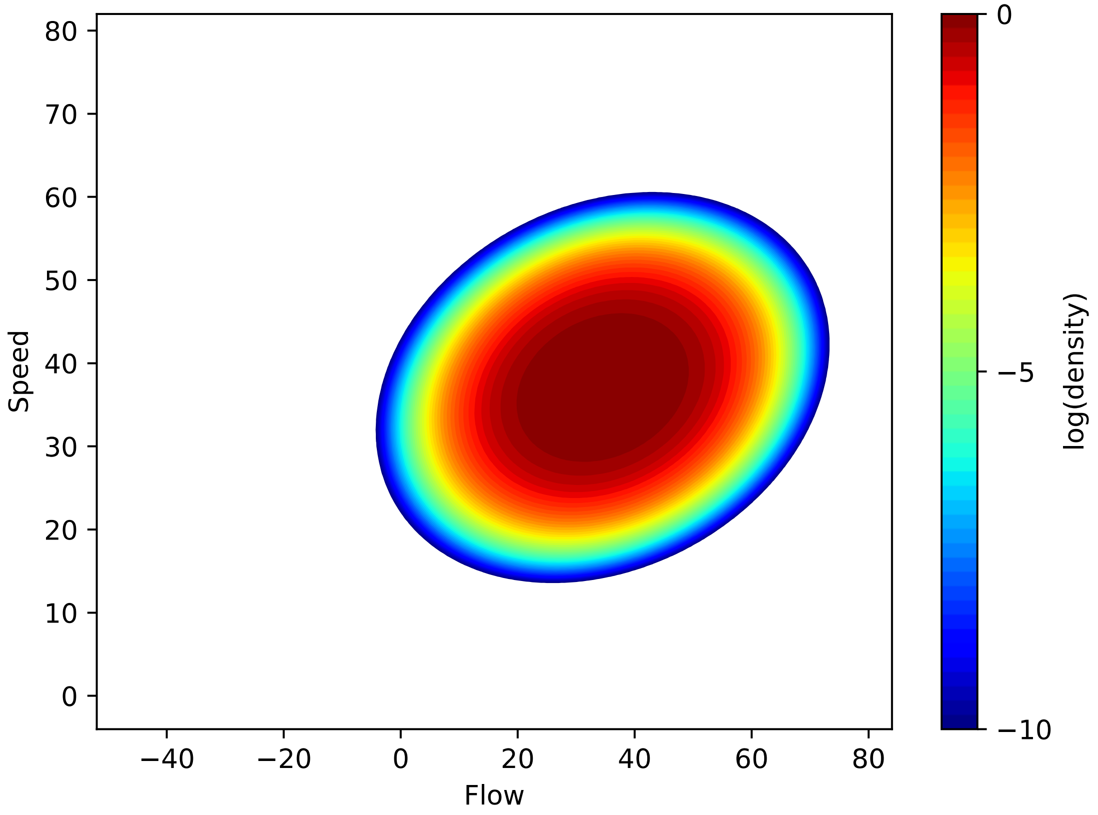


Difficulties in Data Modelling

Data Histogram

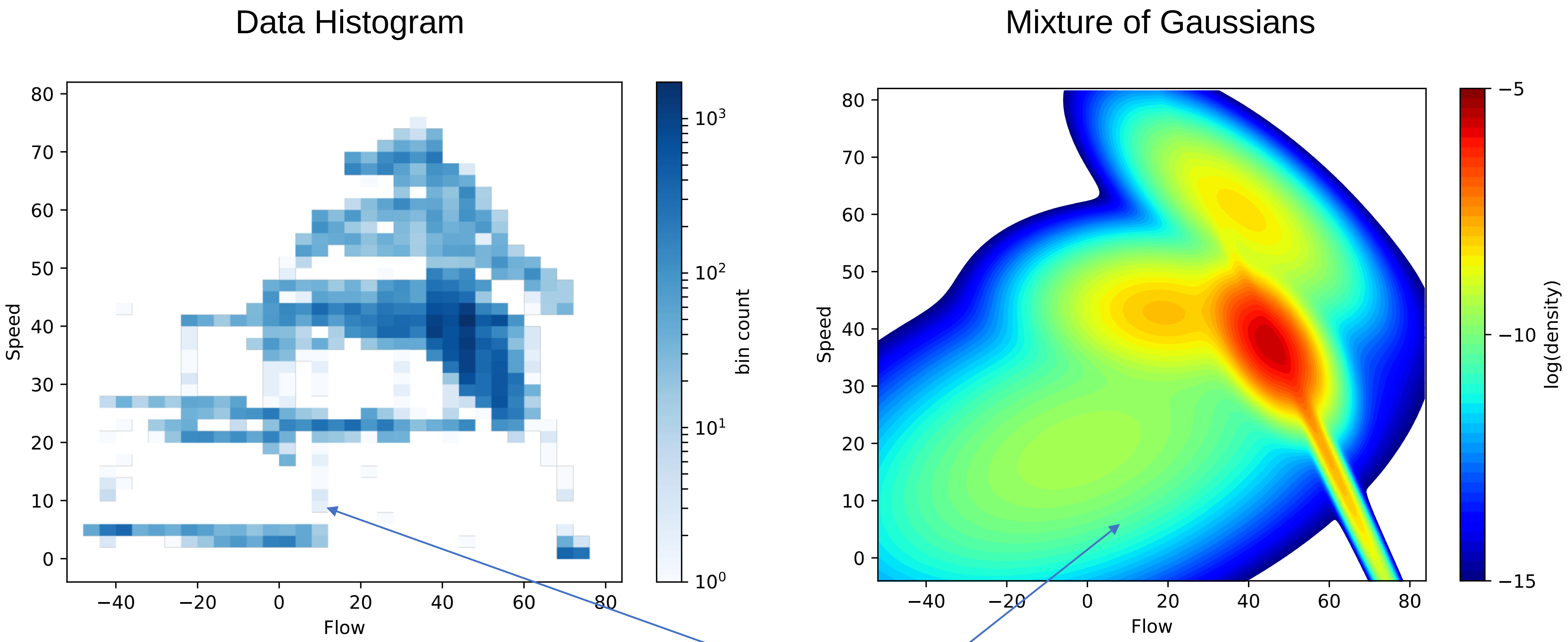


Gaussian Model



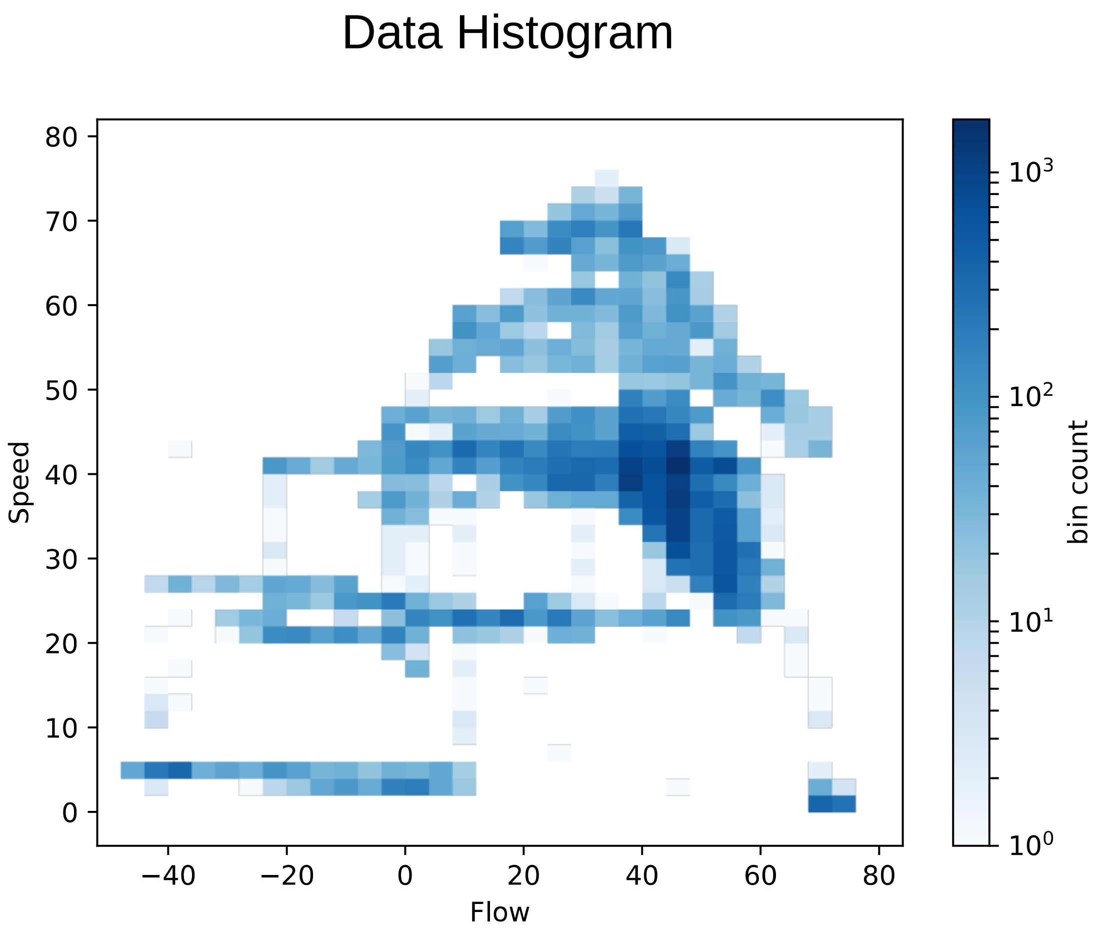
Poor Fit

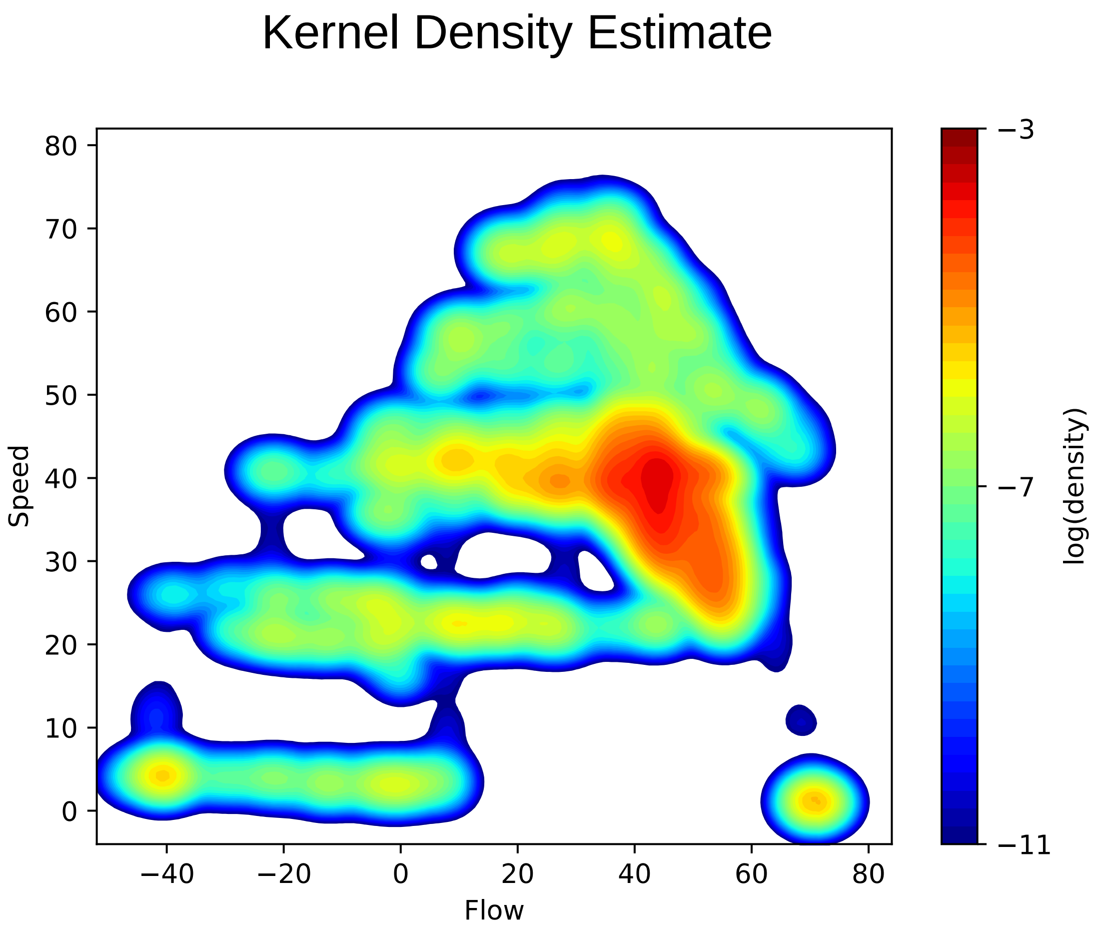
Difficulties in Data Modelling



Inaccurate: Gaps not captured

Kernel Density Estimation (KDE)

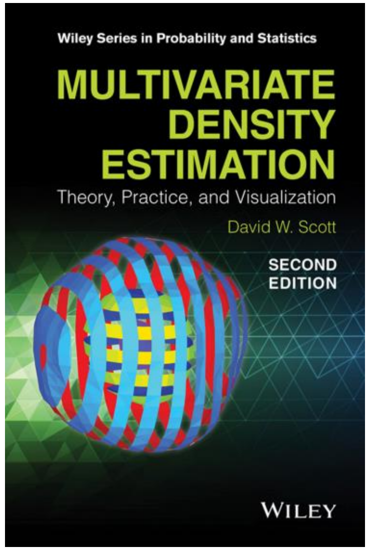


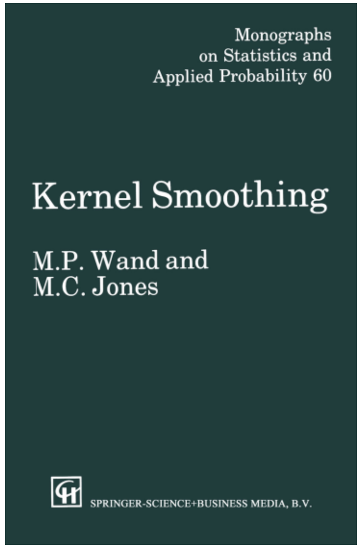


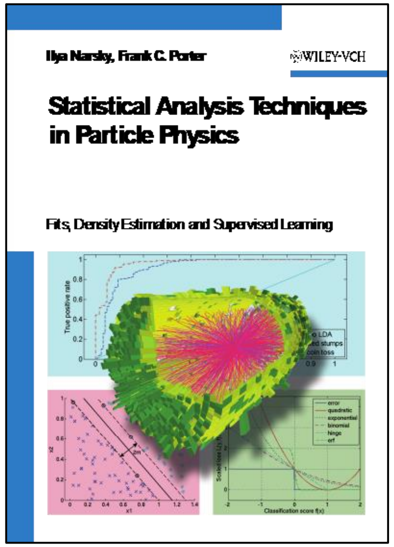
Much better fit

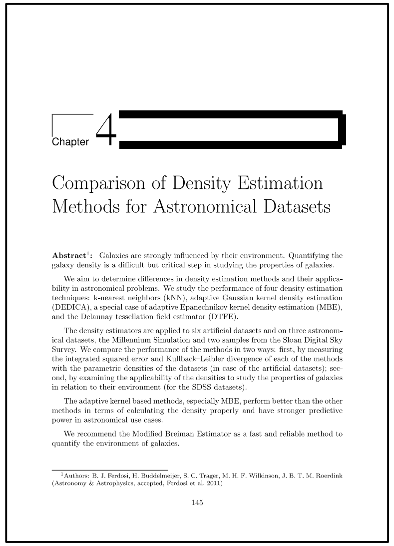
KDE: Statistical Gold Standard

- Guaranteed to converge to the underlying distribution
- Provides normalized, true probability densities
- Few assumptions about shape of distribution: inferred from data









Chapter 4

Comparison of Density Estimation Methods for Astronomical Datasets

Abstract[1]**:** Galaxies are strongly influenced by their environment. Quantifying the galaxy density is a difficult but critical step in studying the properties of galaxies.

We aim to determine differences in density estimation methods and their applicability in astronomical problems. We study the performance of four density estimation techniques: k-nearest neighbors (kNN), adaptive Gaussian kernel density estimation (DEDICA), a special case of adaptive Epanechnikov kernel density estimation (MBE), and the Delaunay tessellation field estimator (DTFE).

The density estimators are applied to six artificial datasets and on three astronomical datasets, the Millennium Simulation and two samples from the Sloan Digital Sky Survey. We compare the performance of the methods in two ways: first, by measuring the integrated squared error and Kullback–Leibler divergence of each of the methods with the parametric densities of the datasets (in case of the artificial datasets); second, by examining the applicability of the densities to study the properties of galaxies in relation to their environment (for the SDSS datasets).

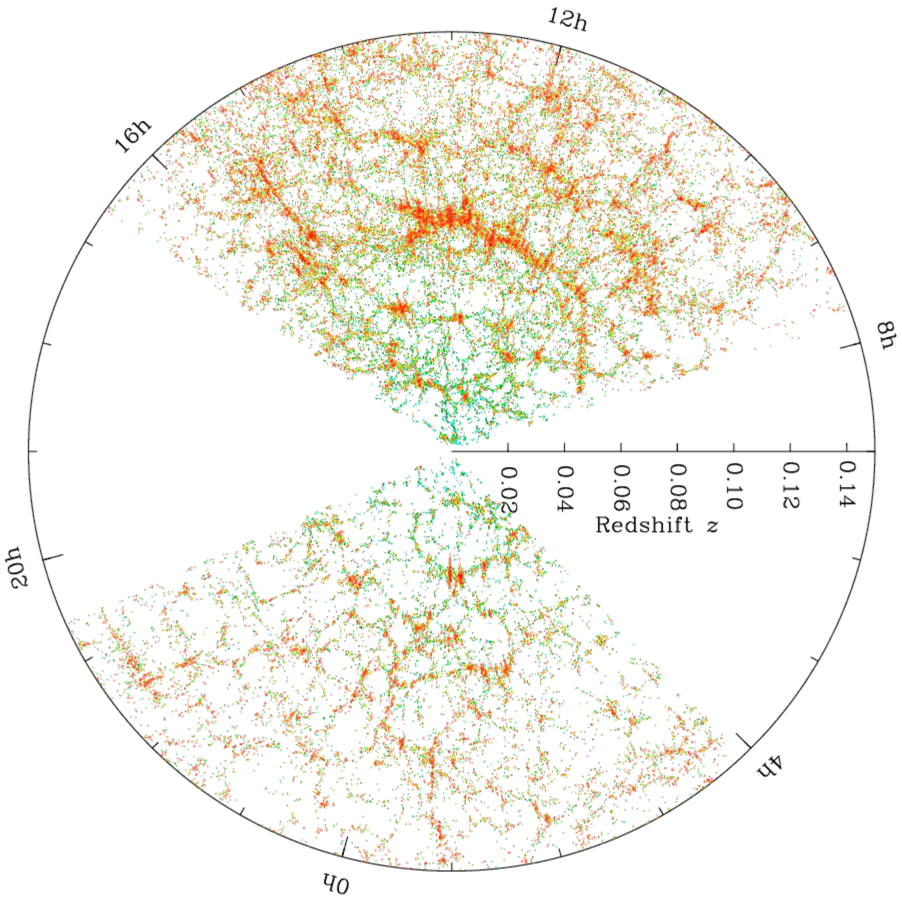
The adaptive kernel based methods, especially MBE, perform better than the other methods in terms of calculating the density properly and have stronger predictive power in astronomical use cases.

We recommend the Modified Breiman Estimator as a fast and reliable method to quantify the environment of galaxies.

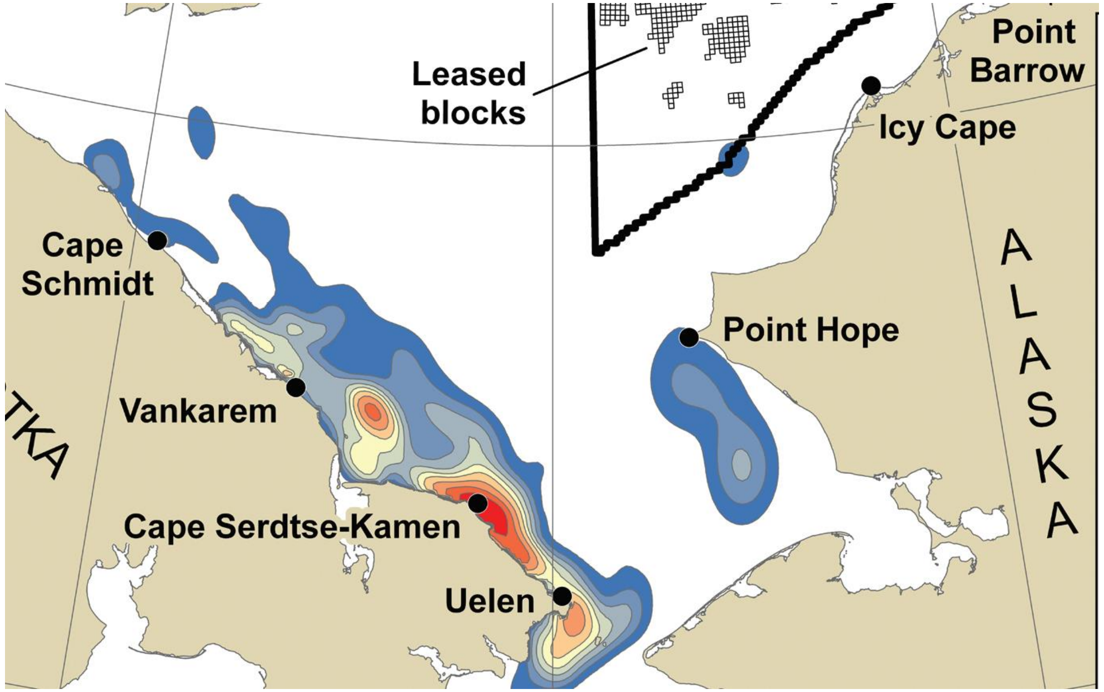
[1]Authors: B. J. Ferdosi, H. Buddelmeijer, S. C. Trager, M. H. F. Wilkinson, J. B. T. M. Roerdink (Astronomy & Astrophysics, accepted, Ferdosi et al. 2011)

145

KDE Usage

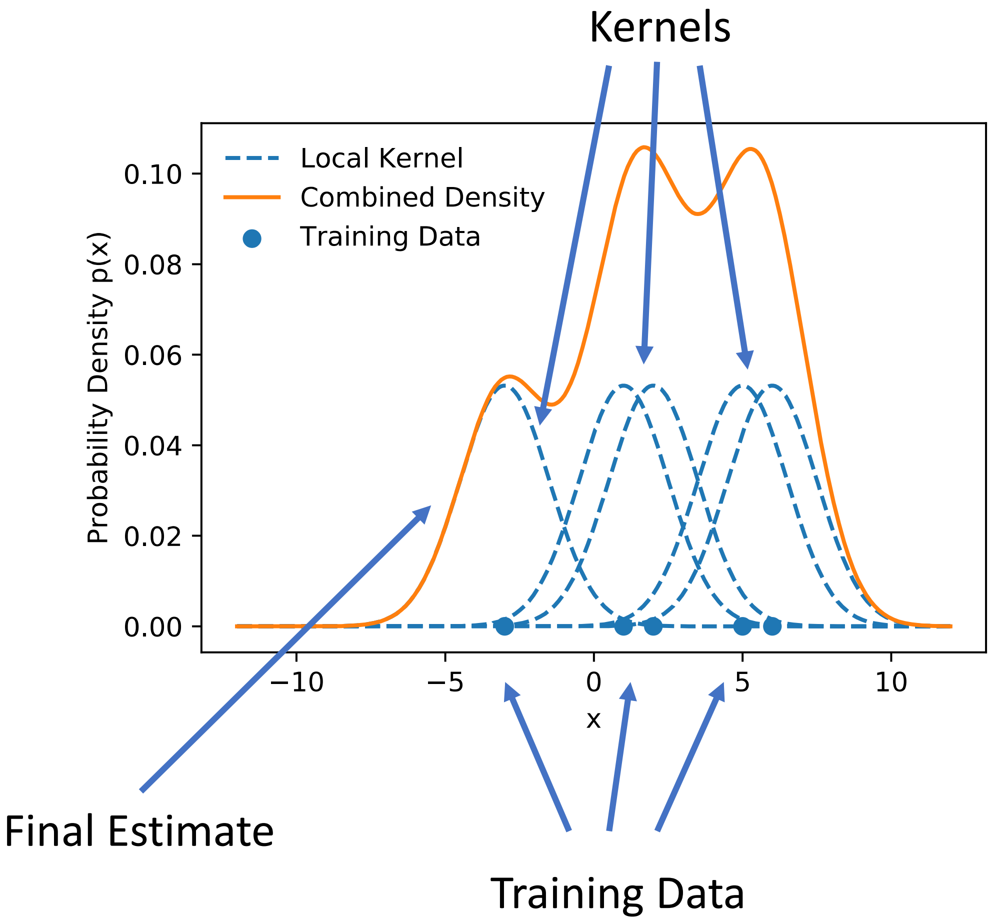


Galaxy Mass Distribution
[Sloan Digital Sky Survey]



Distribution of Bowhead Whales
[L.T. Quackenbush et al, *Arctic* 2010]

KDE Definition



Each point in dataset contributes a *kernel*

Kernel: localized Gaussian “bump”

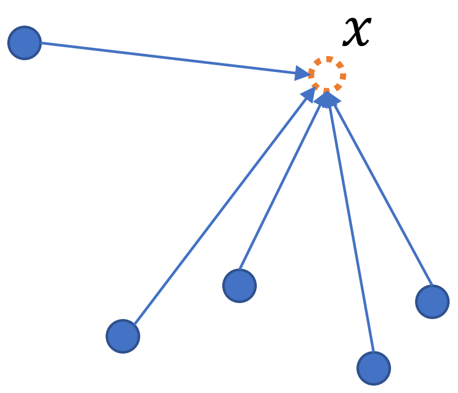
Kernels summed up to form estimate

Mixture of N Gaussians: N is the dataset size

$$f(x) = \frac{1}{n} \sum_{x_i \in \mathrm{Data}} K(x - x_i)$$

Problem: KDE does not scale

$$f\left(x\right)=\frac{1}{n}\sum_{x_i\in\mathrm{Data}} K\left(x-x_i\right)$$



Training Data

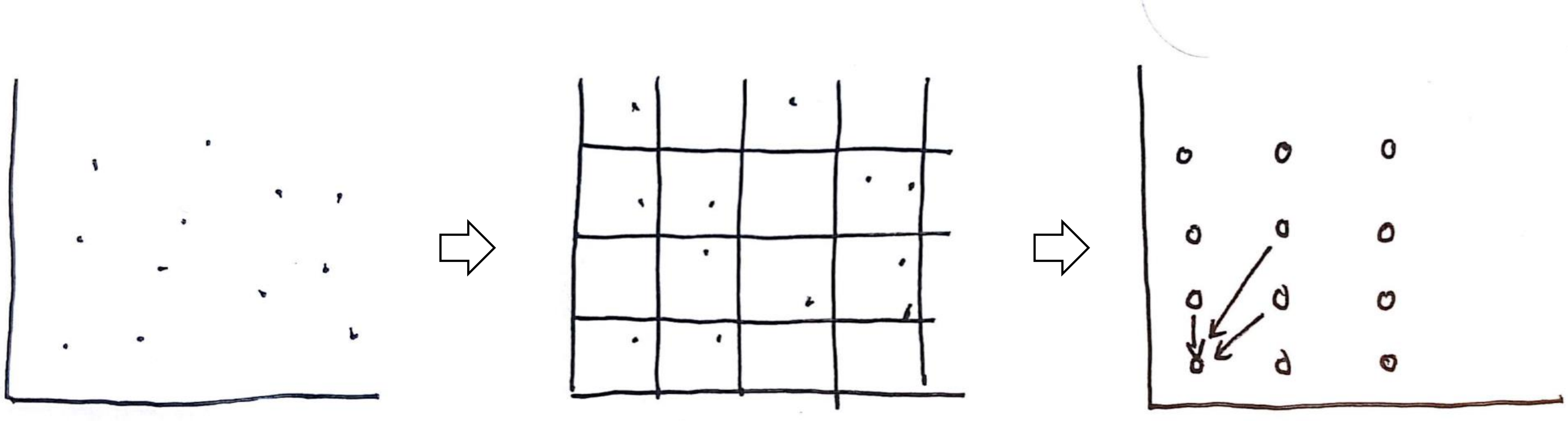
$O(n)$ to compute single density $f(x)$

$O(n^2)$ to compute all densities in data

2 hours to compute on 1M points
on 2.9Ghz Core i5

How can we speed this up?

Strawman Optimization: Histograms



Training Dataset   Binned Counts   Grid computation

Benefit: Runtime depends on grid size rather than N

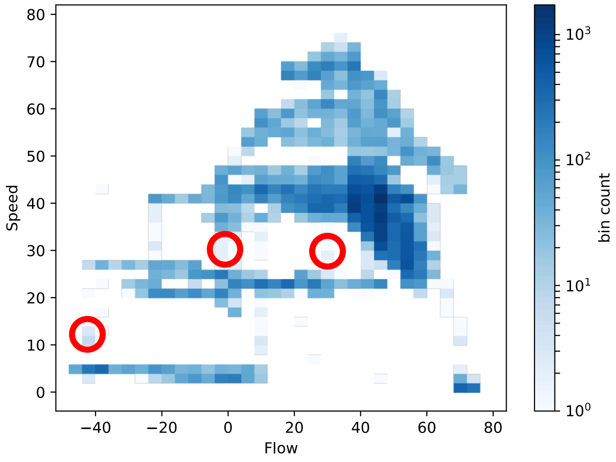
Problem: Bin explosion in high dimensions

[Wand, *J. of Computational and Graphical Statics* 1994]

Stepping Back: What users need

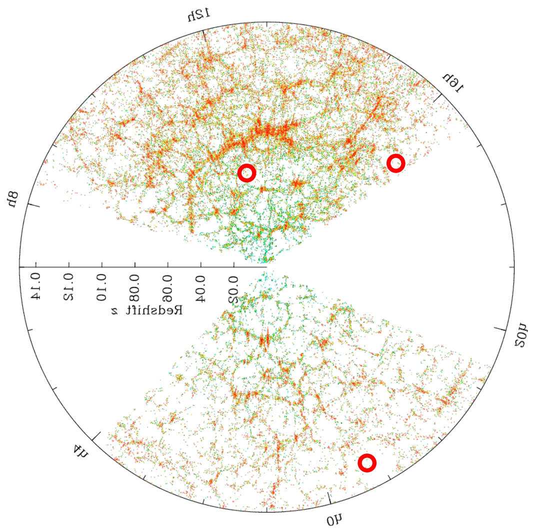
Anomaly Explanation

```
SELECT flight_mode FROM shuttle_sensors
WHERE kde(flow,speed) < threshold
```



Hypothesis Testing

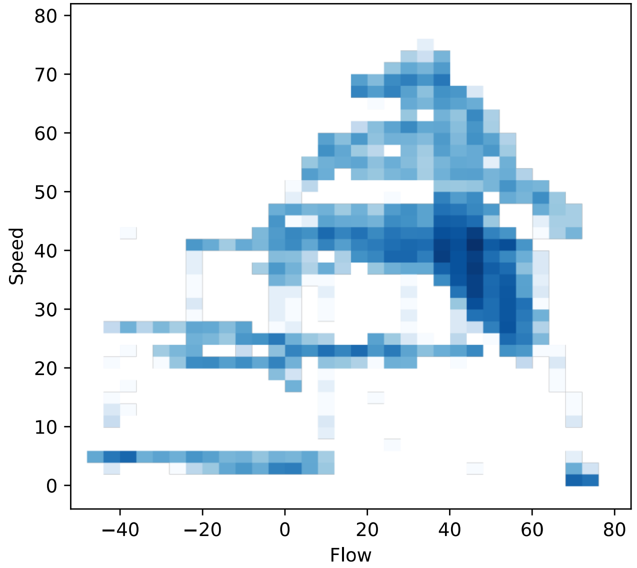
```
SELECT color FROM galaxies
WHERE kde(x,y,z) < threshold
```



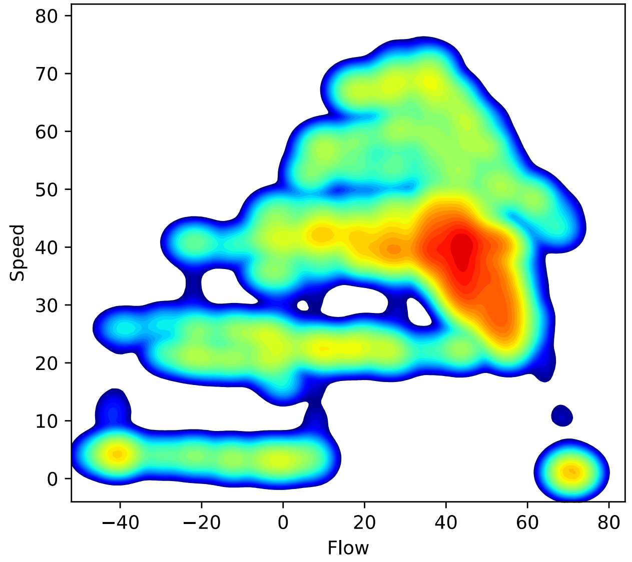
From Estimation to Classification

```
SELECT flight_mode FROM shuttle_sensors
WHERE kde(flow,speed) < Threshold
```

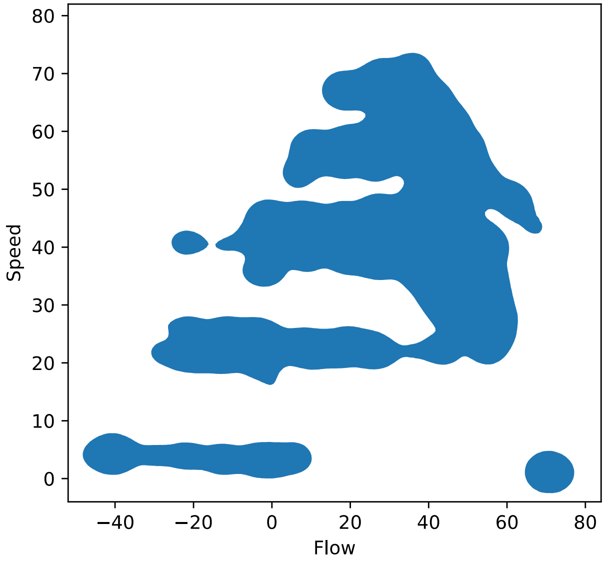
Training Data



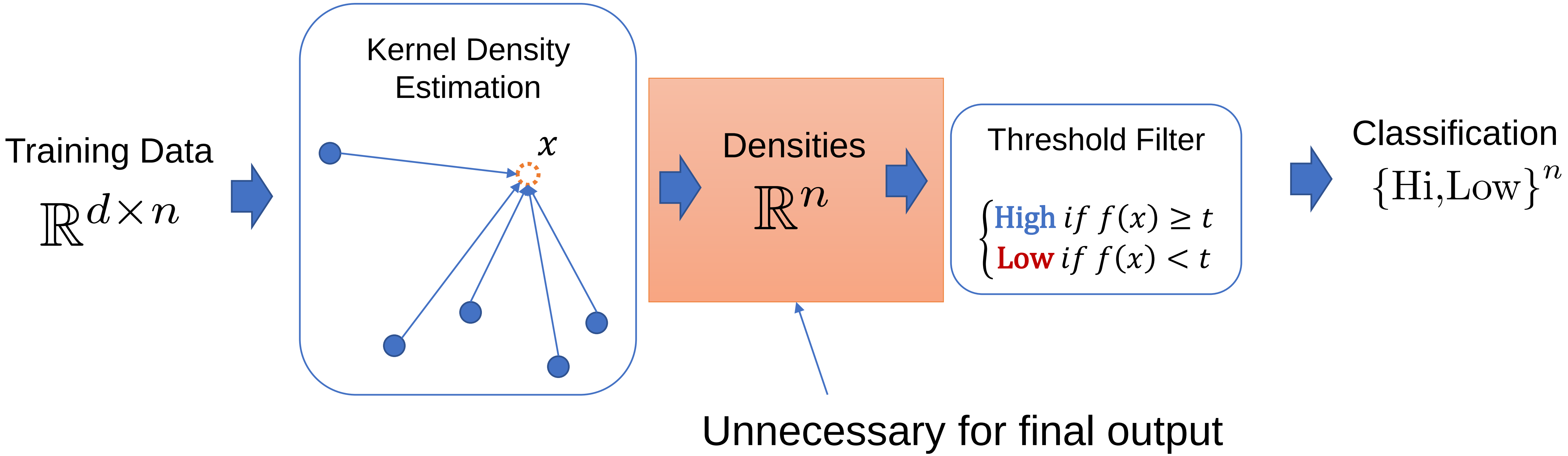
KDE Model



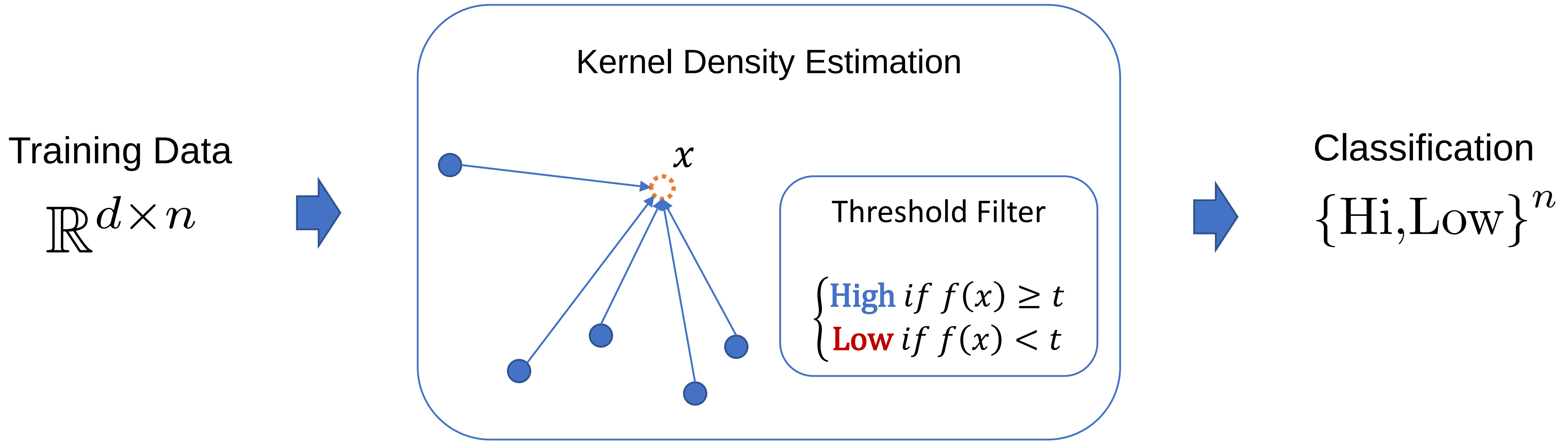
Classification



End to End Query



End to End Query



Recap

- KDE can model complex distributions
- Problem: KDE scales quadratically with dataset size
- Real Usage: KDE + Predicates = Kernel Density Classification
- Idea: Apply Predicate Pushdown to KDE

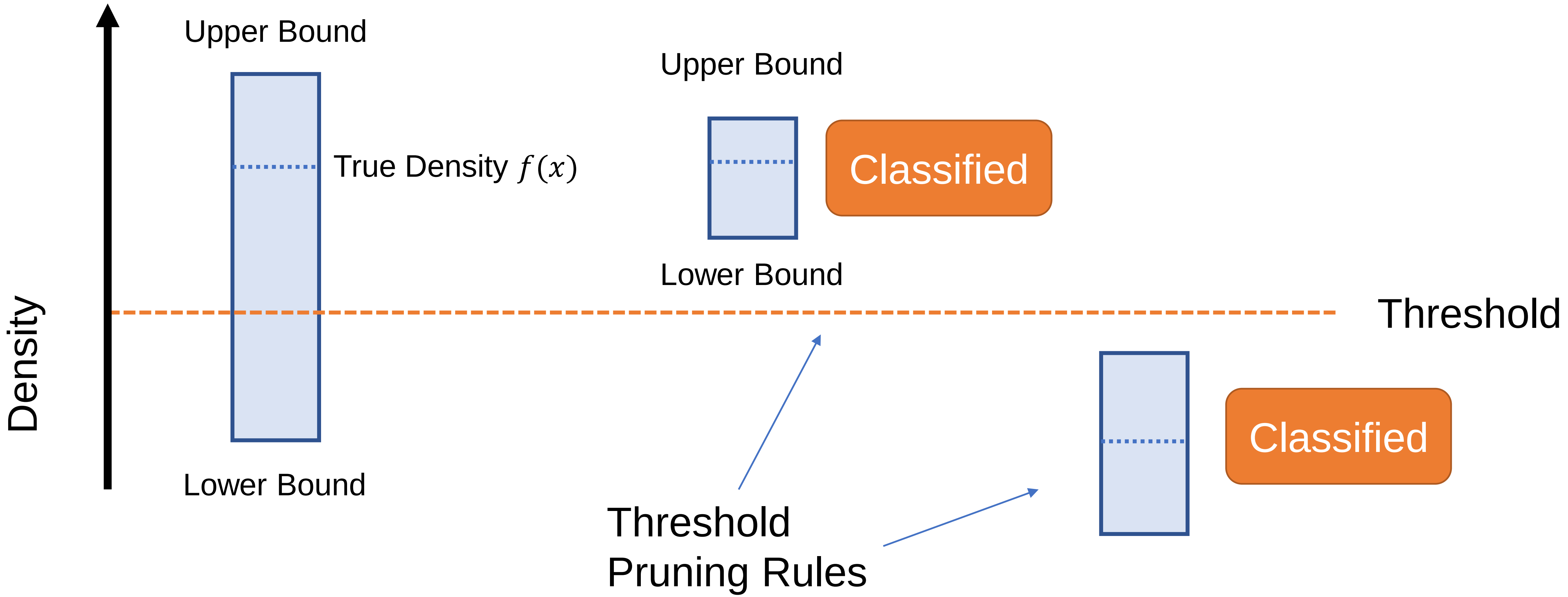
tkdc Algorithm Overview

1. Pick a threshold

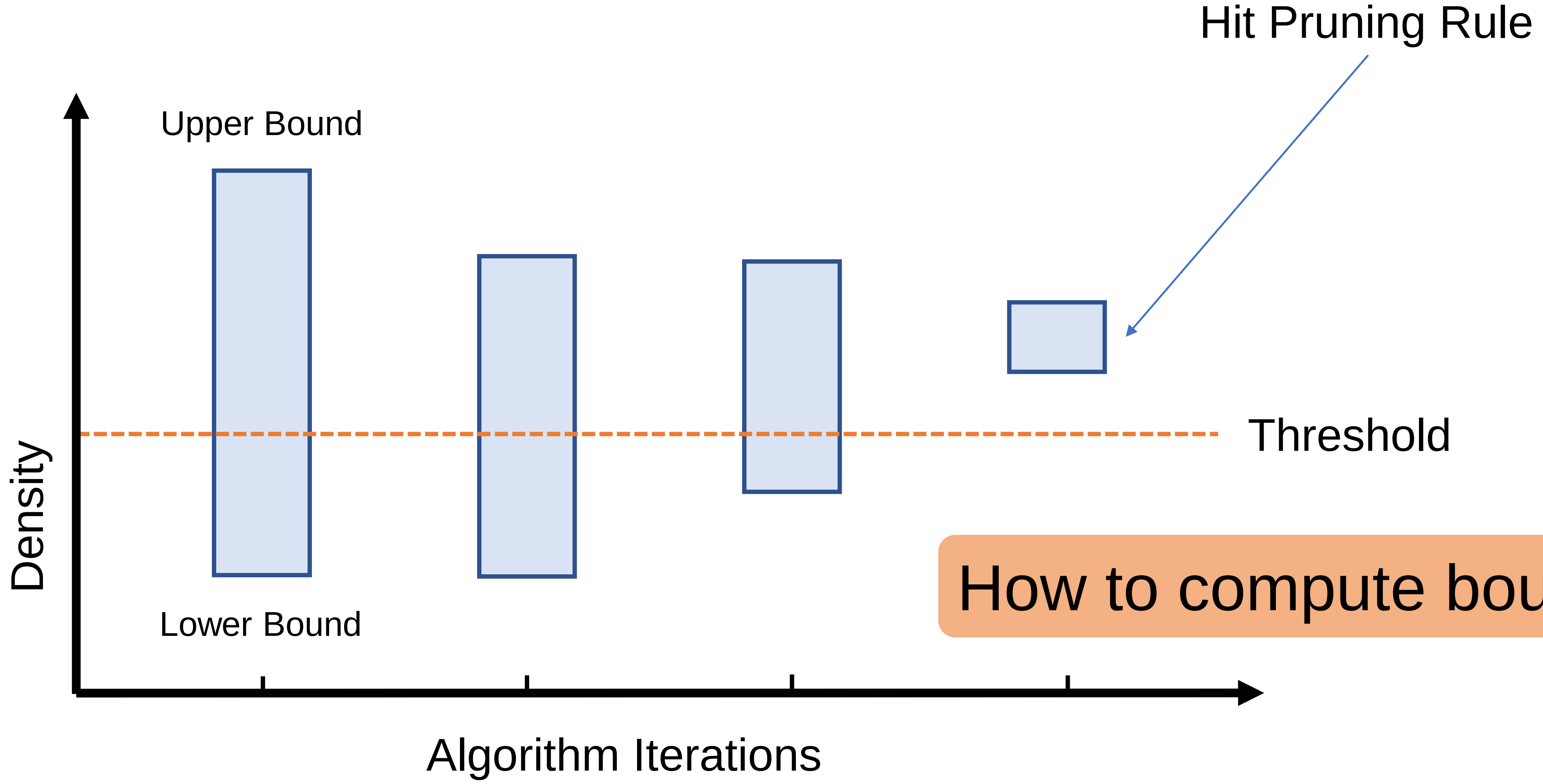
2. Repeat: Calculate bounds on point density

3. Stop when we can make a classification

Classifying the density based on bounds

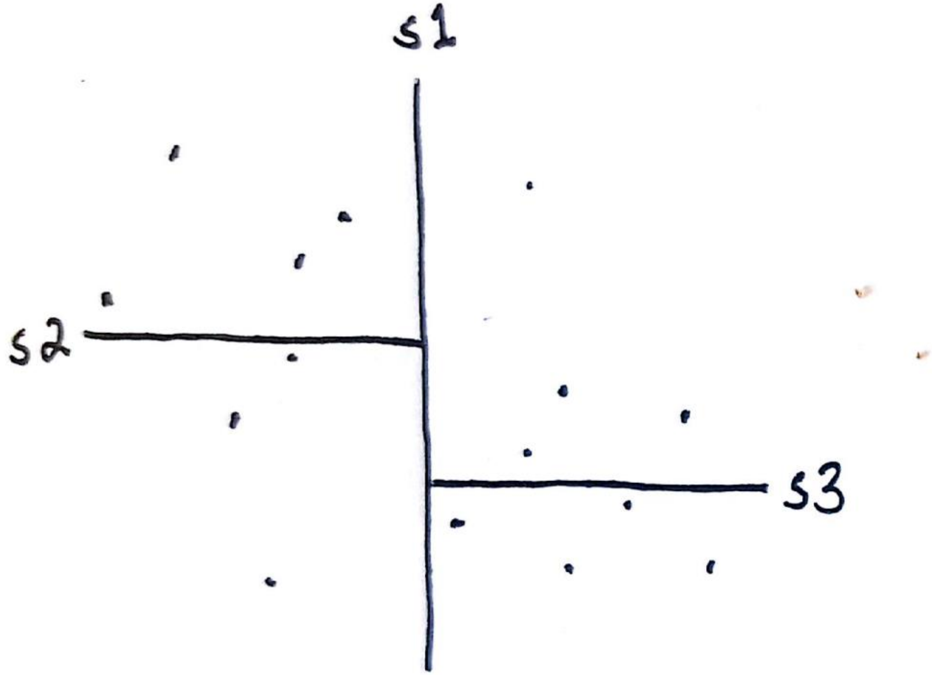


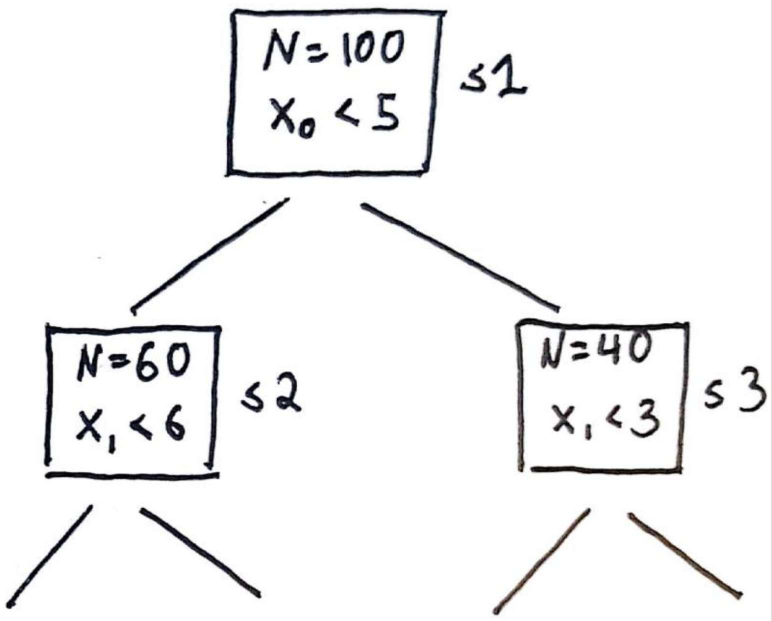
Iterative Refinement



How to compute bounds?

k-d tree Spatial Indices





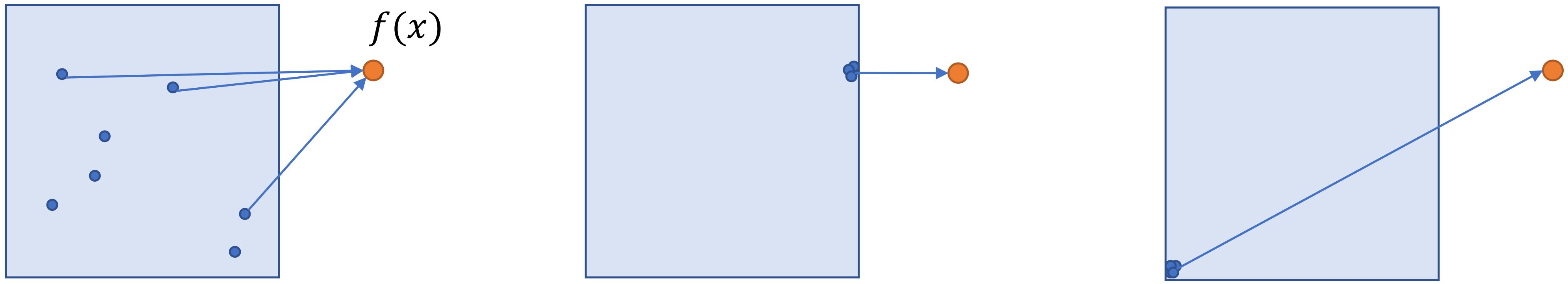
Divide N-dimensional space
1 axis at a time

Nodes for each Region
Track # of points + bounding box

Bounding the densities

Given from k-d tree: Bounding Box, # Points Contained

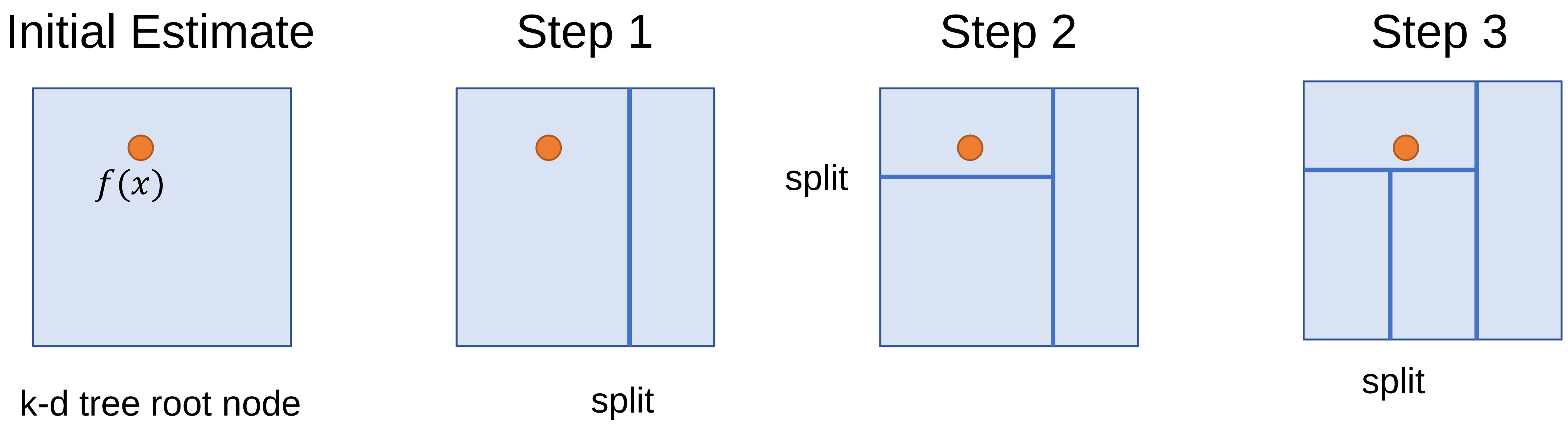
Total contribution from a region can be bounded



Total Contribution   Maximum Contribution   Minimum Contribution

[Gray & Moore, *ICDM* 2003]

Iterative Refinement



Priority Queue: Split nodes with largest uncertainty first

tkdc Algorithm Overview

1. Pick a threshold
   - User-Specified
   - Automatically Inferred

2. Calculate bounds on a density
   - k-d tree bounding boxes

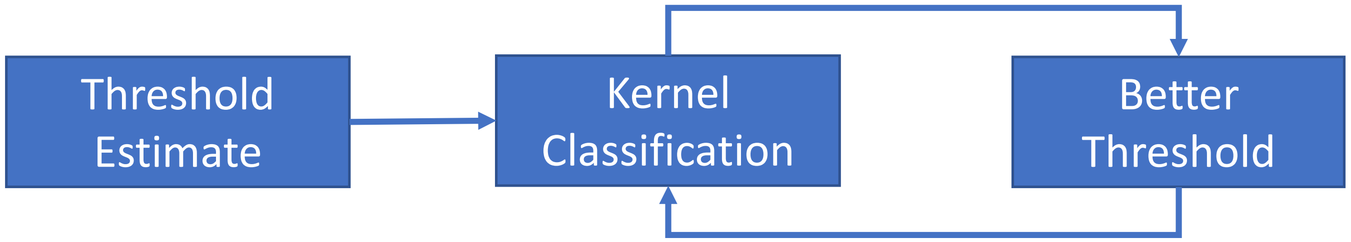
3. Refine the bounds until we can classify
   - Priority-queue guided region splitting

Automatic Threshold Selection

- Probability Densities hard to work with:
  - Unpredictable
  - Huge range of magnitudes
- Good Default: capture a set % of the data

```
SELECT Quantile(kde(A,B), 1%) from shuttle_sensors
```

- Bootstrapping



- Classification for computing thresholds
  - See paper for details

tkdc Complete Algorithm

- Pick a threshold
  - Inferred given desired % level
- Calculate bounds on a density
  - k-d tree bounding boxes
- Refine the bounds until we can make classification
  - Priority-queue guided region splitting

Theorem: Expected Runtime

n number of training points

d dimensionality of data

$$\text{Runtime} = O\left(n^{\frac{d-1}{d}}\right)$$

$$\text{Naive} = O\left(n\right)$$

100 million data points, 2-dimensions $\frac{100M}{(100M)^{\frac{1}{2}}} \approx$ 10,000x

100 million data points, 8-dimensions $\frac{100M}{(100M)^{\frac{7}{8}}} \approx$ 10x

Runtime in practice: Experimental Setup

Single Threaded, In-memory

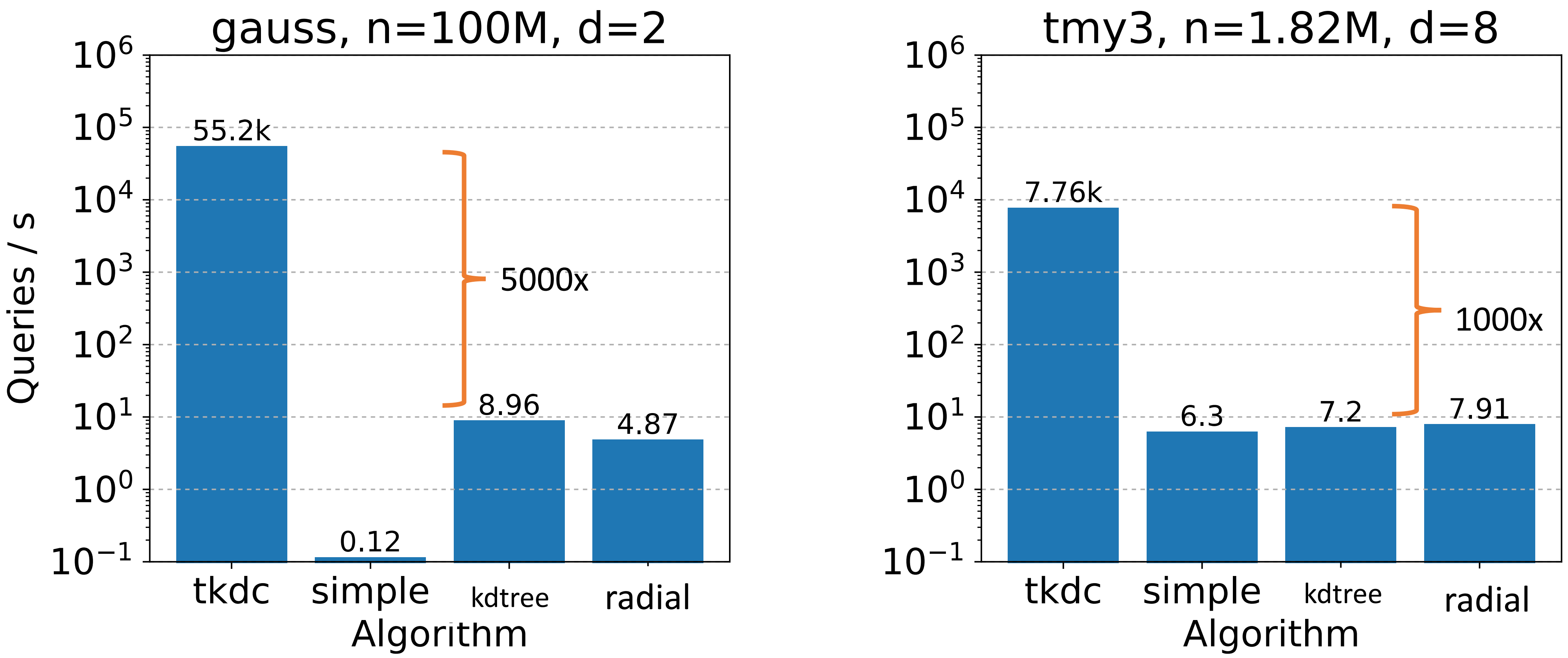
Total Time = Training Time + Threshold Estimation + Classify All

Threshold = 1% classification rate

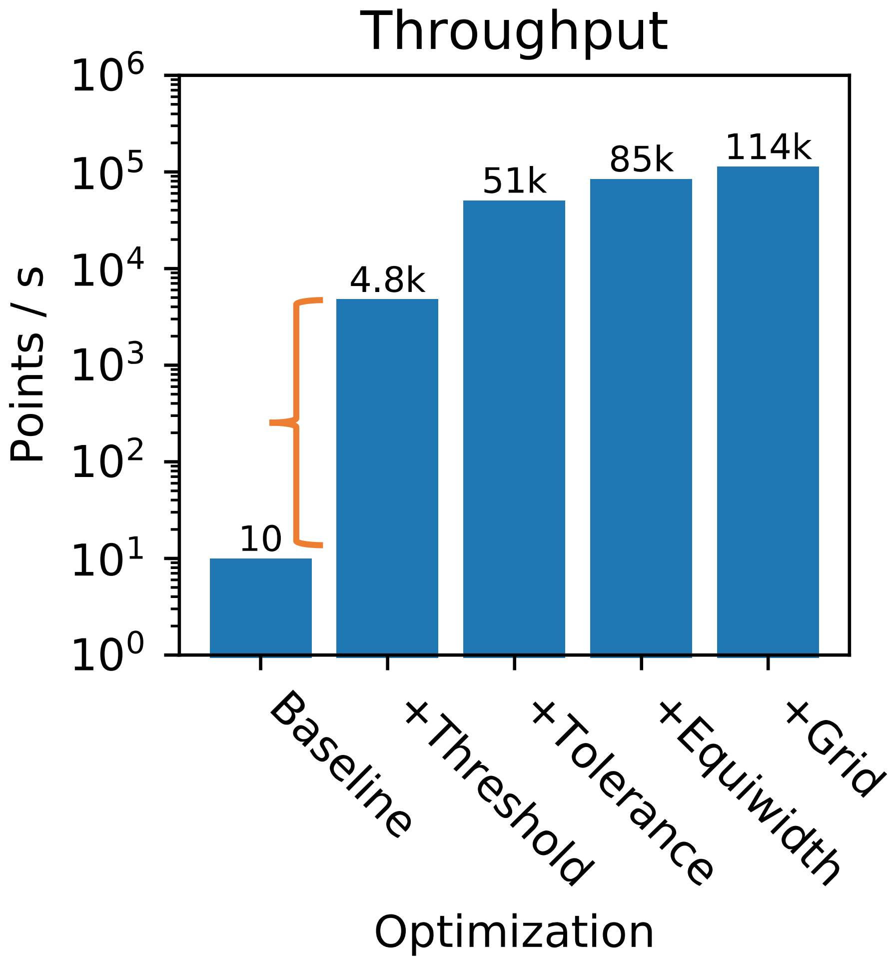
Baselines:

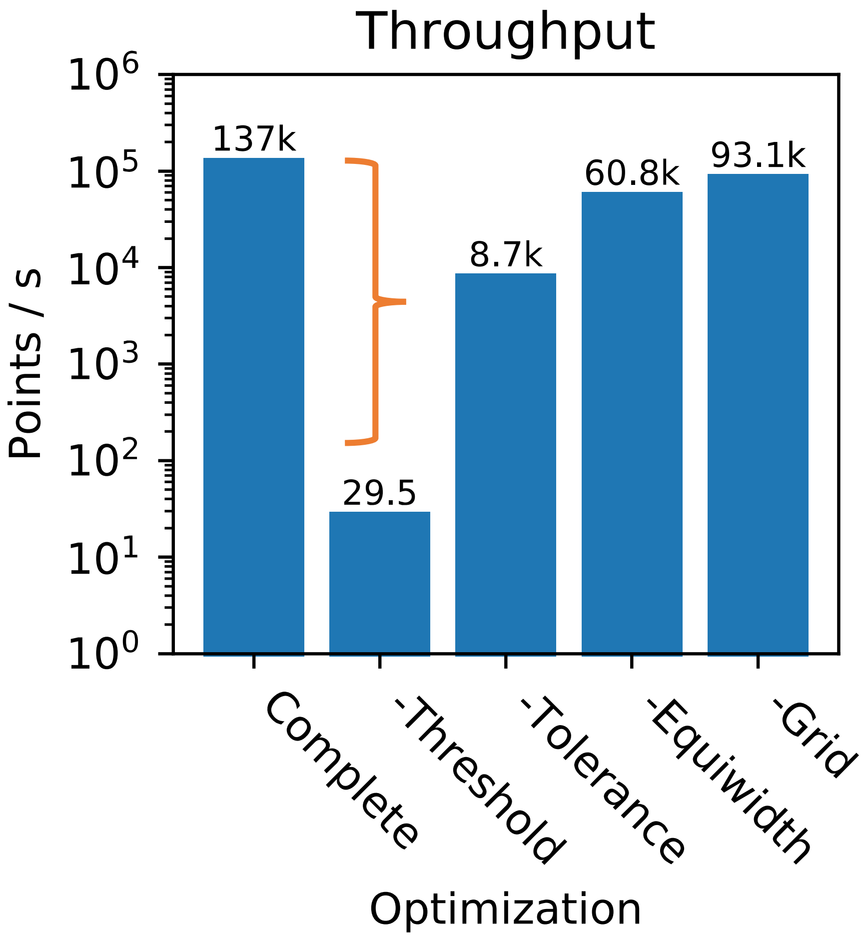
- simple: naïve for loop over all points
- kdtree: k-d tree approximate density estimation, no threshold
- radial: iterates through points, pruning > certain radius

KDE Performance Improvement

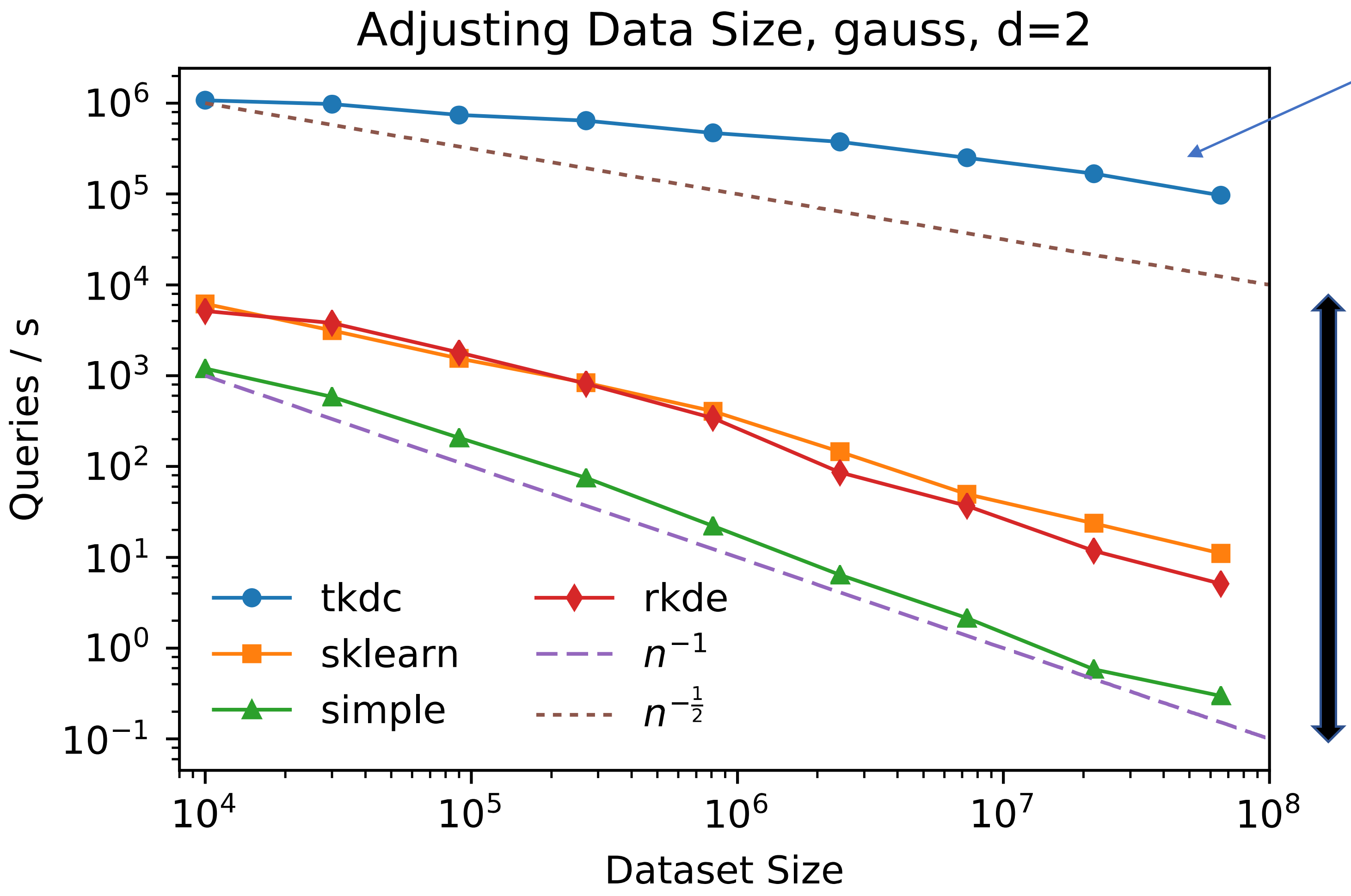


Threshold Pruning Contribution





tkdc scales well with dataset size



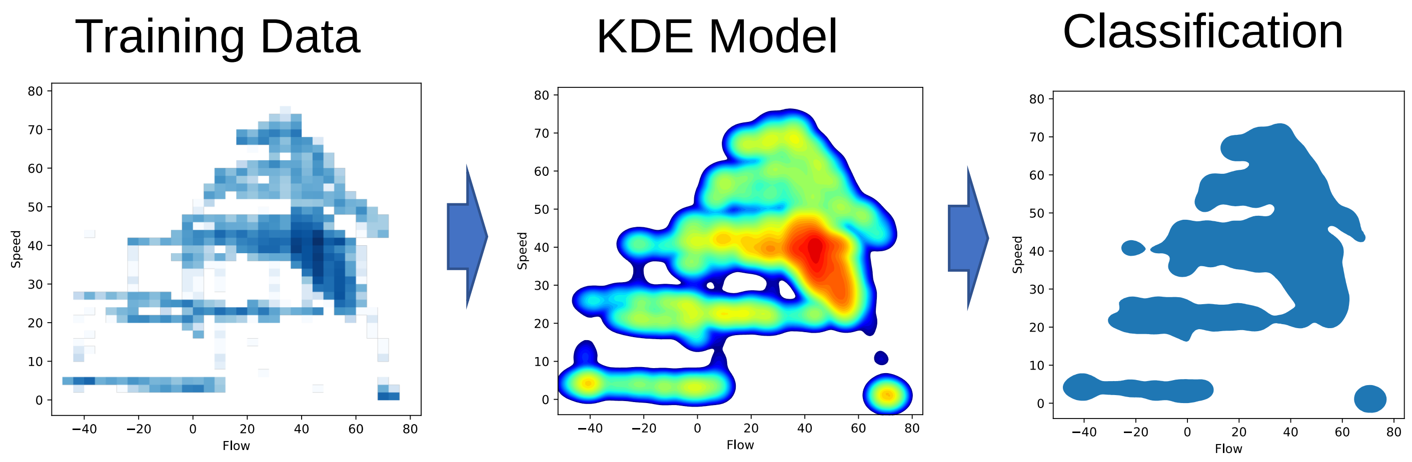
Our Algorithm: tkdc

Asymptotic Speedup

Conclusion

KDE:
Powerful & Expensive

```
SELECT flight_mode FROM shuttle_sensors
WHERE kde(flow, speed) < Threshold
```



Real Queries:
MacroBase

Systems Techniques:

Predicate Pushdown, k-d tree indices:

1000x, Asymptotic Speedups

https://github.com/stanford-futuredata/tKDC