Practical Low-Dimensional Halfspace Range Space Sampling

Michael Matheny

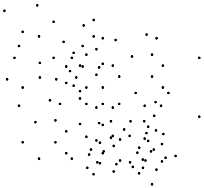
with Jeff Phillips

School of Computing
University of Utah

December 13, 2018

Problem Setup

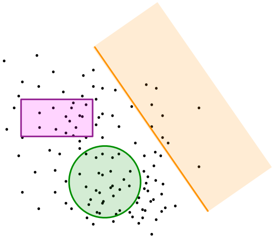
Let X be a set of n points in $\mathbb{R}^d$.



Problem Setup

Let X be a set of n points in $\mathbb{R}^d$.

Let $(X, \mathcal{A})$ be a range space with constant VC dimension, defining a family of subsets $\mathcal{A}$ of X.



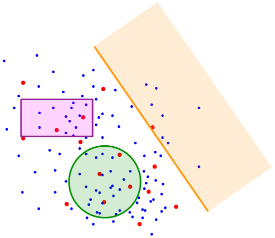
Problem Setup

Let X be a set of n points in $\mathbb{R}^d$.

Let $(X, \mathcal{A})$ be a range space with constant VC dimension, defining a family of subsets $\mathcal{A}$ of X.

Then a subset $S \subseteq X$ is an ε-sample if:

$$\left| \frac{|X \cap A|}{|X|} - \frac{|S \cap A|}{|S|} \right| \leq \varepsilon$$



Motivations

ε-samples can be used as a general preprocessing step for:

- Range searching.
- Spatial Anomalies.
- Discrepancy computation.
- Heat maps.

Motivations(cont)

Small ε-samples can be used to potentially speed up actual real world problems such as spatial scan statistics!

FiveThirtyEight

Politics Sports **Science & Health** Economics Culture

NOV. 16, 2016 AT 8:00 AM

How New York Hunts For Early Signs Of Disease Outbreaks

By Ian Evans

Filed under Public Health

On July 29, 2015, the New York City Department of Health and Mental Hygiene sent out an alert — 31 people in the South Bronx had contracted Legionnaires' disease, a lung infection from waterborne bacteria that kills about 1 out of every 10 people who get it. By the time officials found the source (a cooling tower) and contained the spread, 128 people had contracted Legionnaires' and 12 people had died. It was the largest outbreak of Legionnaires' disease in the city's history — an outbreak that was first detected by a computer program.

Contructing ε-samples

Given a range space $(X, \mathcal{A})$ with VC dimension d then a random sample $S \subseteq X$ with probability $1 - \delta$ will be an ε-sample [VC71, LLS01]

- $|S| = O(\frac{1}{\varepsilon^2}(d + \log \frac{1}{\delta}))$.
- $O(m + \frac{1}{\varepsilon^2}(d + \log \frac{1}{\delta}))$ time.

Sizes of ε-samples

Can do better than random sampling. There exists ε-samples of size:

- Halfspaces $|S| = \Theta(1/\varepsilon^{2d/(d+1)})$ [Mat95].
- Rectangles $|S| = O_d(1/\varepsilon \log^d \frac{1}{\varepsilon})$ [BG17].
- Balls $|S| = O(1/\varepsilon^{2d/(d+1)} \log^{d/(d+1)} \frac{1}{\varepsilon})$ [MWW93].

Sizes of ε-samples

Can do better than random sampling. There exists ε-samples of size:

- **Halfspaces** $|S| = \Theta(1/\varepsilon^{2d/(d+1)})$ [Mat95].
- Rectangles $|S| = O_d(1/\varepsilon \log^d \frac{1}{\varepsilon})$ [BG17].
- Balls $|S| = O(1/\varepsilon^{2d/(d+1)} \log^{d/(d+1)} \frac{1}{\varepsilon})$ [MWW93].

Halfspace ε-samples

At first proofs for smaller sized ε-samples for halfspaces were not constructive [Mat09, Cha00].

- In 2010, Bansal [Ban10] introduced a polynomial time coloring.
  - Runtime of $O(n(1/\varepsilon)^{2d(3d+2)/(d+1)}\mathrm{polylog}(1/\varepsilon))$ [LM15] using merge-reduce framework [CM96].

Halfspace ε-samples

At first proofs for smaller sized ε-samples for halfspaces were not constructive [Mat09, Cha00].

- In 2010, Bansal [Ban10] introduced a polynomial time coloring.
  - Runtime of $O(n(1/\varepsilon)^{2d(3d+2)/(d+1)}\mathrm{polylog}(1/\varepsilon))$ [LM15] using merge-reduce framework [CM96].
  - Can be run on random sample in $O(n + (1/\varepsilon)^{2d(3d+2)/(d+1)+2}\mathrm{polylog}(1/\varepsilon))$ time.

Halfspace ε-samples

At first proofs for smaller sized ε-samples for halfspaces were not constructive [Mat09, Cha00].

- In 2010, Bansal [Ban10] introduced a polynomial time coloring.
  - Runtime of $O(n(1/\varepsilon)^{2d(3d+2)/(d+1)}\mathrm{polylog}(1/\varepsilon))$ [LM15] using merge-reduce framework [CM96].
  - Can be run on random sample in $O(n + (1/\varepsilon)^{2d(3d+2)/(d+1)+2}\mathrm{polylog}(1/\varepsilon))$ time.
  - When $d = 2$ is $O(n + (1/\varepsilon)^{12+2/3}\mathrm{polylog}(1/\varepsilon))$

Halfspace ε-samples

At first proofs for smaller sized ε-samples for halfspaces were not constructive [Mat09, Cha00].

- In 2010, Bansal [Ban10] introduced a polynomial time coloring.
  - Runtime of $O(n(1/\varepsilon)^{2d(3d+2)/(d+1)}\mathrm{polylog}(1/\varepsilon))$ [LM15] using merge-reduce framework [CM96].
  - Can be run on random sample in $O(n + (1/\varepsilon)^{2d(3d+2)/(d+1)+2}\mathrm{polylog}(1/\varepsilon))$ time.
  - When $d = 2$ is $O(n + (1/\varepsilon)^{12+2/3}\mathrm{polylog}(1/\varepsilon))$
- Other constructions with slightly worse size guarantees [STZ04, BCEG07]
  - Most require at least $\Omega(n + (1/\varepsilon^{2d/(d+1)})^d)$ time.

Halfspace ε-samples

At first proofs for smaller sized ε-samples for halfspaces were not constructive [Mat09, Cha00].

- In 2010, Bansal [Ban10] introduced a polynomial time coloring.
  - Runtime of $O(n(1/\varepsilon)^{2d(3d+2)/(d+1)}\mathrm{polylog}(1/\varepsilon))$ [LM15] using merge-reduce framework [CM96].
  - Can be run on random sample in $O(n + (1/\varepsilon)^{2d(3d+2)/(d+1)+2}\mathrm{polylog}(1/\varepsilon))$ time.
  - When $d = 2$ is $O(n + (1/\varepsilon)^{12+2/3}\mathrm{polylog}(1/\varepsilon))$
- Other constructions with slightly worse size guarantees [STZ04, BCEG07]
  - Most require at least $\Omega(n + (1/\varepsilon^{2d/(d+1)})^d)$ time.
- Some methods with same guarantee as random sampling, but work better in practice [HAM06]

Halfspace ε-samples (cont)

We want a method with similar performance as random sampling's $O(n + \frac{1}{\varepsilon^2})$ time and similar size to the optimal $\Theta(1/\varepsilon^{2d/(d+1)})$.

Halfspace ε-samples (cont)

We want a method with similar performance as random sampling's $O(n + \frac{1}{\varepsilon^2})$ time and similar size to the optimal $\Theta(1/\varepsilon^{2d/(d+1)})$.

We present a simple method with

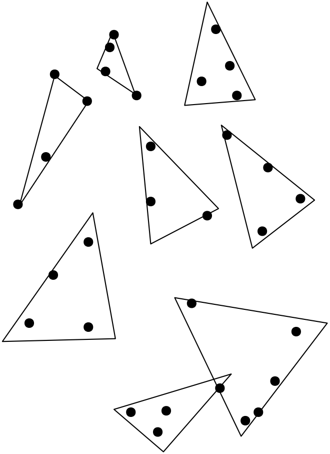
- $|S| = O((1/\varepsilon)^{2d/(d+1)} \log^{d/(d+1)}(1/\varepsilon))$
- Computable in $O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$ time.

Experimental results.

Partition

A *partition* of $(X, \mathcal{H}_d)$

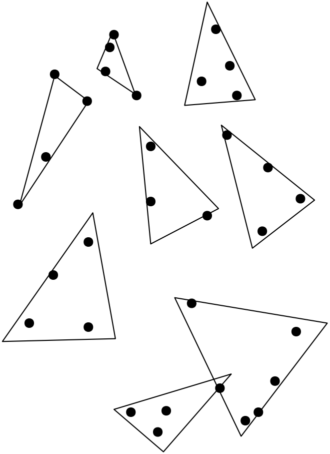
- Pairs $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.
- $X_i \subseteq \Delta_i \cap X$ and $X_i \cap X_j = \emptyset$.
- In a (t, z)-*partition* there are $O(t)$ pairs, $|X_i| \leq 2n/t$; and each $h \in \mathcal{H}_d$ crosses $O(t^z)$ cells.



Partition

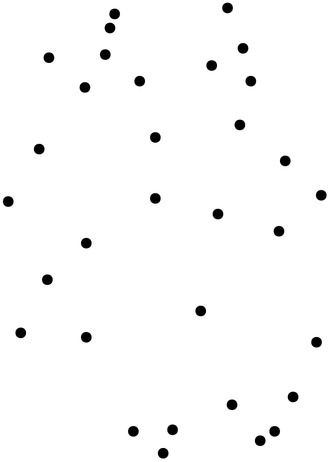
A *partition* of $(X, \mathcal{H}_d)$

- Pairs $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.
- $X_i \subseteq \Delta_i \cap X$ and $X_i \cap X_j = \emptyset$.
- In a (t, z)-*partition* there are $O(t)$ pairs, $|X_i| \leq 2n/t$; and each $h \in \mathcal{H}_d$ crosses $O(t^z)$ cells.
- At best $z = 1 - 1/d$
  - $z = .5$ when $d = 2$



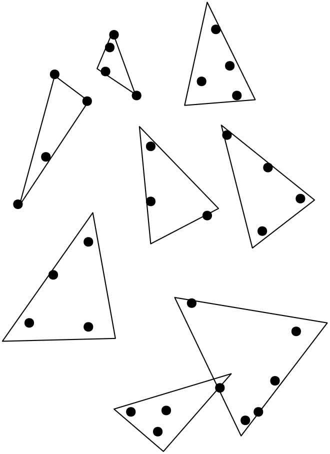
Algorithm

- Points X.



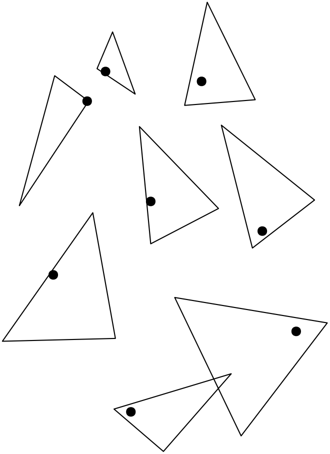
Algorithm

- Points X.
- Construct partitioning $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.



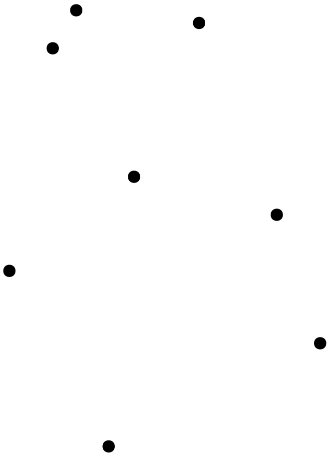
Algorithm

- Points X.
- Construct partitioning $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.
- Sample a random point p_i from X_i and assign weight $w_i = |X_i|$.



Algorithm

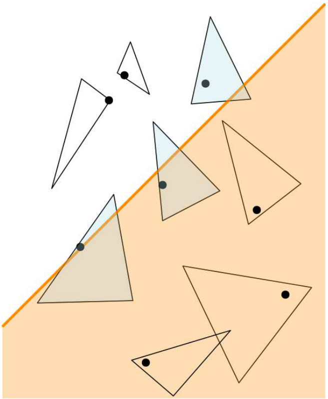
- Points X.
- Construct partitioning $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.
- Sample a random point p_i from X_i and assign weight $w_i = |X_i|$.
- Output is a weighted sample S.



Algorithm

Partitioning of m points into t partitions can be done in $O(m \log t)$ time with $z = 1 - 1/d$ [Cha10].

- Running partitioning on a random sample $m = O(\frac{1}{\varepsilon^2})$ with constant probability.
- Time is $O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$.

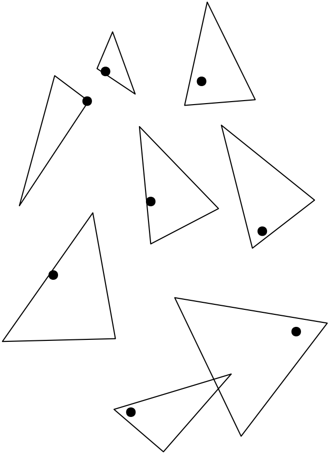


Formal Statement

For range space $(X, \mathcal{H}_d)$ with $|X| = n$ and constant d, with constant probability an ε-sample S of size $O(\frac{1}{\varepsilon^{2d/(d+1)}} \log^{d/(d+1)} \frac{1}{\varepsilon})$ can be constructed in $O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$ time.

Algorithm

- Points X.
- **Construct partitioning** $\{(\Delta_1, X_1), (\Delta_2, X_2), \ldots\}$.
- Sample a random point p_i from X_i and assign weight $w_i = |X_i|$.
- Output is a weighted sample S.



Implementation Details

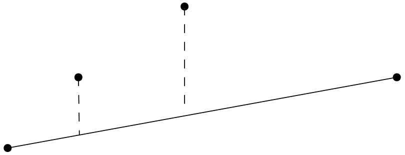
Several sampling algorithms in python for $d = 2$.

- Matousek's efficient partition trees [Mat92], $z = .5$.
- Chan's optimal partition trees [Cha10], $z = .5$.
  - Full implementation, Chan.
  - Simpler (less optimal) implementation, Chan Simple.
- Ham Tree Sample [Wil82] and Double Ham Tree [EW86] with $z = .792$ and $z = .695$.
- Biased-L2 [HAM06]
  - Does not rely on partitioning.
  - Guarantee is no better than random sampling.

Implementation Details (cont)

From bottom to top:

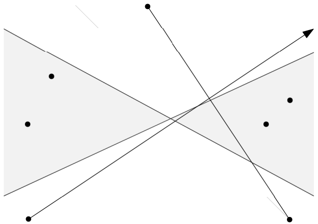
- Line and point primitives.

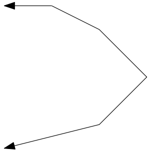


Implementation Details (cont)

From bottom to top:

- Line and point primitives.
- Line segments, wedges, and polygons.

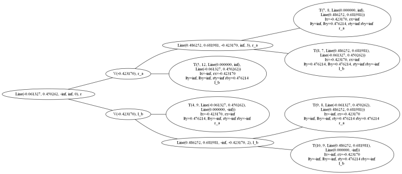


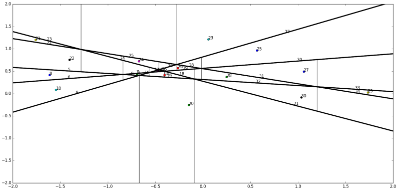


Implementation Details (cont)

From bottom to top:

- Line and point primitives.
- Line segments, wedges, and polygons.
- PolyTree to keep track of an arrangement [Sei91, HP00].

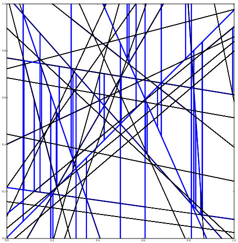


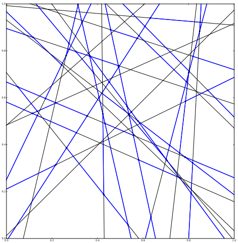


Implementation Details (cont)

From bottom to top:

- Line and point primitives.
- Line segments, wedges, and polygons.
- PolyTree to keep track of an arrangement [Sei91, HP00].
- Cuttings, approximate ham-sandwich cuts, and various queries (intersection).

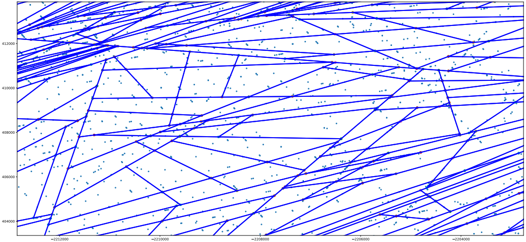




Implementation Details (cont)

From bottom to top:

- Line and point primitives.
- Line segments, wedges, and polygons.
- PolyTree to keep track of an arrangement [Sei91, HP00].
- Cuttings, approximate ham-sandwich cuts, and various queries (intersection).
- Partitioning.



Experimental Setup

Evaluating the sampling.

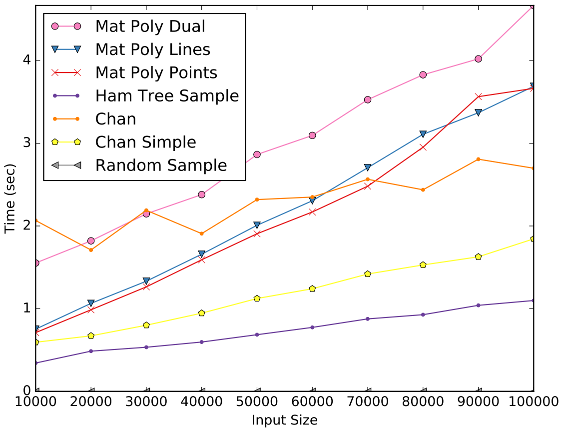
- Took samples on Chicago crime data [Chi17] with 6.5 million data points.
- Evaluate halfplane discrepancy on resulting sample S.
  - $\text{Error}(X, S) = \max_{h \in \mathcal{H}_d} \left| \frac{|S \cap h|}{|S|} - \frac{|X \cap h|}{|X|} \right|$.
- Vary sample size and compare with random sampling.

Experimental Results

We found that Biased-L2 [HAM06] was extremely slow.

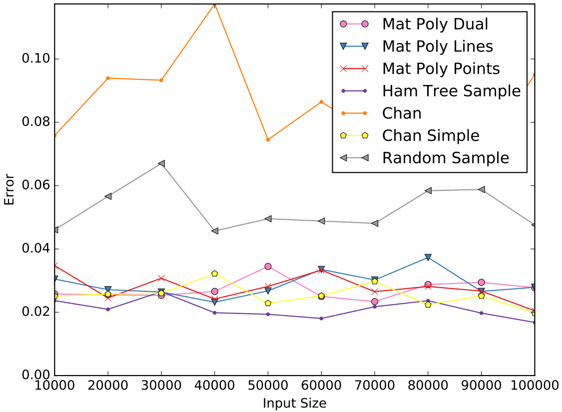
Experimental Results

- Computation time increases roughly linearly with the input size.
- Random sampling is much faster.



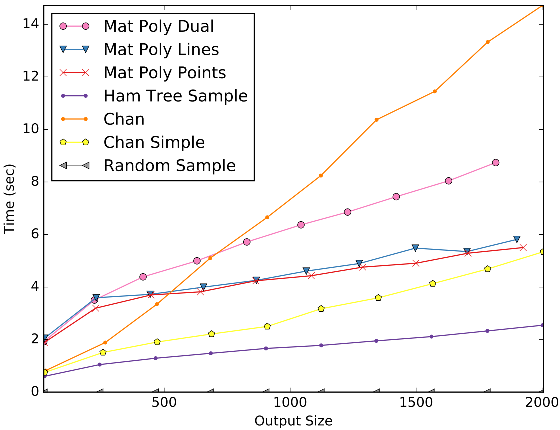
Experimental Results

- Error remains relatively constant with input size.



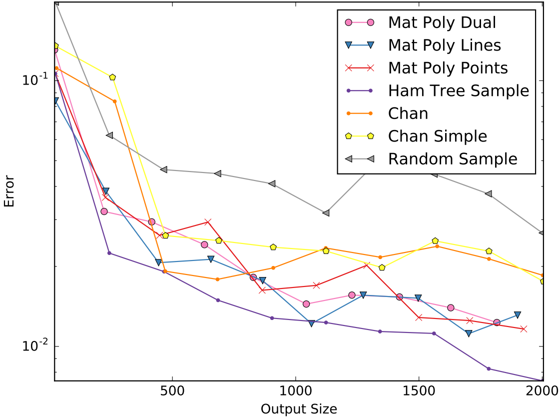
Experimental Results

- Time increases with output size in a roughly linear fashion.
- Chan has a much higher constant factor.



Experimental Results

- All partitioning methods produce significantly smaller samples than random sampling.
- Ham Tree Sample is by far the best even with its larger $z = .792$.



Applications

Use these sampling method for finding approximate discrepancy (can also be used for scan statistics).

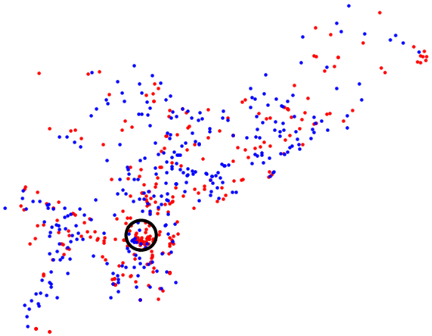


Figure: Philadelphia crime data for vehicular theft.

Faster with better coreset bounds?

Approximate discrepancy is computable in

$$O(n + \frac{1}{\varepsilon}|S| \log \frac{1}{\varepsilon} + T(n, |S|))$$

time, where $|S|$ is the ε-sample size and $T(n, k)$ its construction time.

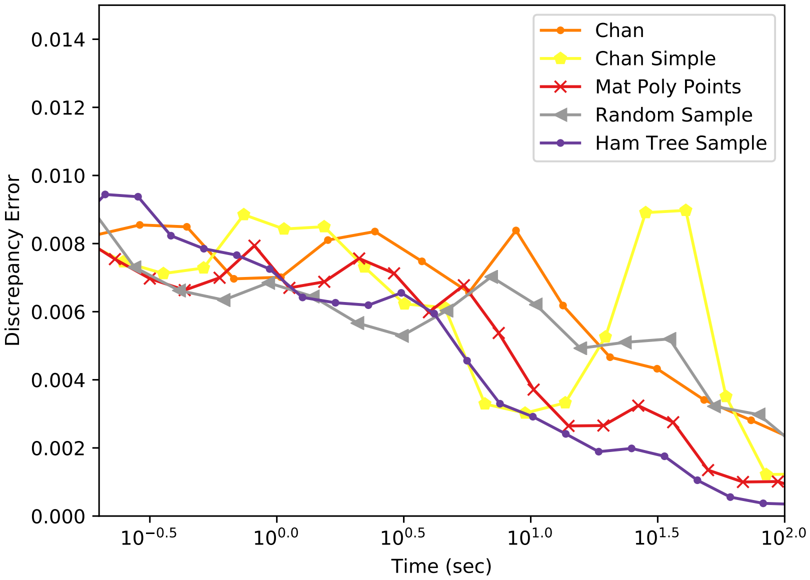
- Halfplane Scanning with Chan

$$O(n + \frac{1}{\varepsilon^{2+\frac{1}{3}}} \log^{1+\frac{2}{3}} \frac{1}{\varepsilon})$$

- Halfplane Scanning with random sampling

$$O(n + \frac{1}{\varepsilon^3} \log \frac{1}{\varepsilon})$$

Runtimes



Chan
Chan Simple
Mat Poly Points
Random Sample
Ham Tree Sample
Discrepancy Error
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
$10^{-0.5}$
$10^{0.0}$
$10^{0.5}$
$10^{1.0}$
$10^{1.5}$
$10^{2.0}$
Time (sec)

Takeaways

- Best method
  - Ham Tree Sample and Double Ham Tree work well in practice and are simple to implement.
  - Better theoretical methods could be useful at very large scales.
  - Different methods can be composed.
- If the points are uniformly distributed then even a *kd*-tree will give an optimal $z = \frac{1}{2}$ partitioning [Mat94].
- This method can probably be used for polynomials by using polynomial partitioning.

For Further Reading I

- Nikhil Bansal, *Constructive algorithms for discrepancy minimization*, Proceedings 51st Annual IEEE Symposium on Foundations of Computer Science, 2010, pp. 407–414.
- Amitabha Bagchi, Amitabh Chaudhary, David Eppstein, and Michael T. Goodrich, *Deterministic sampling and range counting in geometric data streams*, ACM Transactions on Algorithms **3** (2007), no. A16.
- Nikhil Bansal and Shashwat Garg, *Algorithmic discrepancy beyond partial coloring*, Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (New York, NY, USA), STOC 2017, ACM, 2017, pp. 914–926.
- Bernard Chazelle, *The discrepancy method*, Cambridge, 2000.
- Timothy M. Chan, *Optimal partition trees*, In: Proc. 26th Annu. ACM Sympos. Comput. Geom, 2010, pp. 1–10.
- *Crimes in Chicago*, `https://www.kaggle.com/currie32/crimes-in-chicago`, 2017.

For Further Reading II

- Bernard Chazelle and Jiri Matousek, *On linear-time deterministic algorithms for optimization problems in fixed dimensions*, Journal of Algorithms **21** (1996), 579–597.
- Herbert Edelsbrunner and Emo Welzl, *Halfplanar range search in linear space and o(n0.695) query time*, Information Processing Letters **23** (1986), no. 5, 289 – 293.
- Herve Bronnimann Huseyin Akcan and Robert Marini, *Practical and efficient geometric ϵ-approximations*, Proceedings of the 18th Canadian Conference on Computational Geometry (2006), 120 – 125.
- S. Har-Peled, *Constructing planar cuttings in theory and practice*, SIAM J. Comput. **29** (2000), no. 6, 2016–2039.
- Yi Li, Philip M. Long, and Aravind Srinivasan, *Improved bounds on the samples complexity of learning*, J. Comp. and Sys. Sci. **62** (2001), 516–527.
- Sachar Lovett and Raghu Meka, *Constructive discrepancy minimization by walking on the edges*, SIAM Journal on Computing **44** (2015), 1573–1582.

For Further Reading III

- Jiri Matousek, *Efficient partition trees*, Discrete & Computational Geometry **8** (1992), 315–334.
- ______, *Geometric range searching*, ACM Comput. Surv. **26** (1994), no. 4, 422–461.
- ______, *Tight upper bounds for the discrepancy of halfspaces*, Discrete and Computational Geometry **13** (1995), 593–601.
- ______, *Geometric discrepancy*, Springer, 2009.
- Jiri Matousek, Emo Welzl, and Lorenz Wernisch, *Discrepancy and approximations for bounded VC-dimension*, Combinatorica **13** (1993), 455–466.
- Raimund Seidel, *A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons*, Computational Geometry **1** (1991), 51 – 64.
- Subhash Suri, Csaba D. Tóth, and Yunhong Zhou, *Range counting over multidimensional data streams*, Proceedings 20th Symposium on Computational Geometry, 2004, pp. 160–169.

For Further Reading IV

Vladimir Vapnik and Alexey Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theo. of Prob and App **16** (1971), 264–280.

D. E. Willard, *Polygon retrieval*, SIAM Journal of Computing (11, ed.), 1982, pp. 149–165.