Approximate Clustering Algorithms for High Dimensional Streaming and Distributed Data

A Dissertation Defense by:
Lee A. Carraher

Department of Electrical Engineering and Computing Systems
University of Cincinnati

November 12, 2017

Agenda

- Classification Problem
- Data Analysis and Clustering
- Research Goals Motivation and Hypothesis
- Related Work and Background
- RPHash and Streaming RPHash Algorithms
- Experimental Setup, First comparisons
- Adaptive LSH and Tree-Walk Improvement
- Experiments with TWRP
- Conclusions and Future Directions

Problem Introduction

- Animals evolved to classify
- classification tasks for survival
    - edible, predator, suitable mate
- Can't experience everything
- Instead need models based on attributes
    - teeth size?, leaf shape?, is an engineer?
- Humans are good classifiers for some things

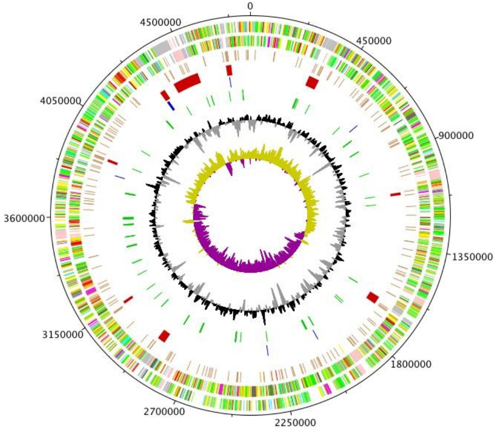
Classifiers

We are good at identifying people.

Classifiers

Not good at large global networking tasks.

Classifiers



Or abstract data clustering tasks.

Classifiers

- We are good at classify things we experience
- Not good at classifying numerical data
- Not good at classifying large data
  - # groups, # attributes
- Not good at classifying unfamiliar things (EKG, Subatomic Interactions...)

Example Problem

A researcher wants to track the geographic dependence or independence, of a particular medical condition. However the patient data reside in disconnected, separately managed databases. The data contains a wide variety of symptoms and biometric measures, for millions of patients. There is an additional constrain placed on the data due to its private and personally identifying nature.

- grouping the symptoms and referencing location could prove or disprove this claim
- high dimensional data
- geographically separated data
- private individually identifiable data

Data Analysis

UNIVERSITY OF Cincinnati

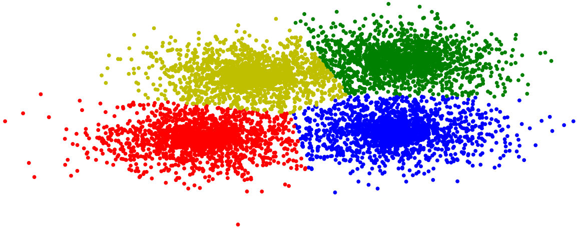
- Data Analysis can provide insights into data that often go beyond human cognition
  - Data with complex relationships (financial trans.)
  - Things foreign to us (subatomic, galactic objects)
  - Lots of fast moving things (tweets)
- How do we create models for these things?

Data Clustering

- Data clustering builds models for classification
- Cluster Models capture similarities between observation attributes
- Clustering is a classic Machine Learning Problem
- The k-Means Problem is a clustering problem

k-means Problem

4-means



- partition observations into *k* subsets
- group observations with similar attributes
- an optimization problem: maximize some objective either inter-cluster or intra-cluster

UNIVERSITY OF Cincinnati

Theorem (Clustering Objective Function)

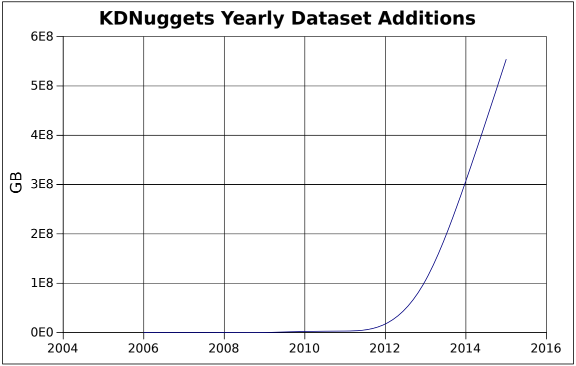
$$\underset{C}{argmin}\sum^{k}\sum_{x\in C}||x-\mu_i||^2$$

- **k-Means** $\in$**Max-SNP**$\in$**APX**$\in$**NP**,
- **APX** are approximable by **PTAS**

Growth of Big Data

Data sets are growing fast

- Medical, Scientific, Financial, Social, Security



- Some are ∞(unbounded) $\rightarrow$ streaming data model.

What needs to be solved?

- PTAS is good complexity for small problems
- Big Data problems require lower complexity
- Streaming algorithms put addition requirements on space
- If we distribute data, are the connections trustworthy?
  - Need security

Goals

- A distributable *k*-means clustering algorithm.
- A streaming *k*-means clustering algorithm.
- Compare clustering performance to other algorithms.
- Show a sub-quadratic complexity growth.
- Evaluate dataset security.
- Experiment with approximate components of RPHash

* same as the proposal

Motivation of RPHash

RPHash is a degenerate case of Locality Sensitive Hash (LSH) based *cr*-NN

- *cr*-NN: given a query x return the k nearest neighbors

Definition (Nearest Neighbor)

[**?**]
Given a set of vectors P in R^d and query vector q return k vectors $p \subseteq P$ such that $p = \text{Argmin}\{dist(p', q)\}$, where dist is some metric function.

LSH NN

Build the DB:

forall the $x \in X$ **do**
  id = LSH_Hash(x)
  T[id].add(x)
end
Return: T

Query the DB:

id = LSH_Hash(q)
Return: linear_scan(T[id],k)

LSH NN

- LSH NN: LSH to narrow search
- Works well in practice!
- Not so well if LSH buckets contain many points
- But can be viewed as centroid candidates

Hypothesis

- Degenerate LSH is useful for Clustering
- Combine Random Projection with LSH gives us an approximate dense region sampling
- Approximate Clustering is equivalent to local minima clustering

Related Work

- Standard algorithm classes
  - *k*-means(Lloyd), mean-shift, agglomerative, spectral
- tend to have bad scalability complexity
- Focus on *k*-means(Lloyd) type
- Tree Based Clustering [**?**]

Scalable Clustering Related Work

UNIVERSITY OF Cincinnati

- Projection based clustering
  - Proclus - project to lower dim.
  - Cluster Ensemble - Histograms
- Density scanning clustering algorithms
  - DBScan
  - Clique
  - CLARANS

Stream Clustering Related Work

- CSketch - most similar
- Streaming k-Means - similar structure
- Damped Sliding Window
- DStream
- Biased Reservoir Sampling

Algorithm Outline

In this thesis we present 3 algorithms for large scale distributed data clustering, that stem from the same basic setup.

- Random Projection Hash (RPHash)
- Streaming RPHash
- Tree-Walk RPHash (TWRP)

RPHash Components

- Random Projection
- Locality Sensitive Hashing
- Exact and approximate counting
- Off-line clustering

Random Projection

UNIVERSITY OF Cincinnati

- Use Random Projection to mitigate Curse of Dimensionality and enforce data embeddings
- JL-Lemma $d \approx \Omega\left(\frac{log(m)}{\epsilon^2 log(1/\epsilon)}\right)$
- d can be even lower for clustering tasks (Bartal '11)
- bonus, unrecoverable data anonymization (Liu,Kargupta,Ryan '06)
- Gaussian RP, DB friendly, and FJLT Projection

LSH Function

Definition (Locality Sensitive Hash Function)

let $\mathbb{H} = \{h : S \to U\}$ is $(r_1, r_2, p_1, p_2)-$sensitive if for any $u, v \in S$

1. if $d(u, v) \leq r_1$ then $Pr_{\mathbb{H}}[h(u) = h(v)] \geq p_1$
2. if $d(u, v) > r_2$ then $Pr_{\mathbb{H}}[h(u) = h(v)] \leq p_2$

- We increase selectivity by appending LSH function results
- And decrease prob. of missing NN by probing many times

Real World LSH

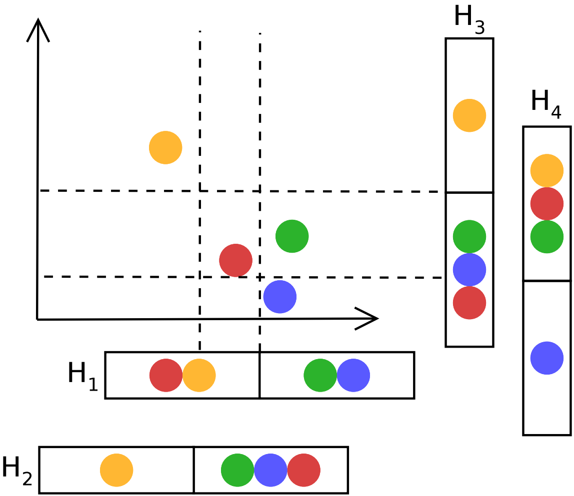
Lattice Based LSH

- Leech Lattice (Λ_{24}) - optimal regular lattice partitioning in $\mathbb{R}^{24}$
- E_8 - optimal regular lattice partitioning in $\mathbb{R}^8$
- D_n - checkerboard lattice

Other LSH

- Spherical - force data to lie on the partitioned surface of a hypersphere
- P-stable - partition orthogonal projections of a vector with a stable distribution
- k-means - use the dual of k-means centroids to define the space partitioning

LSH Based Clustering



Data Prerequisites

- k - number of clusters
- $X = \{x_1, \ldots, x_n\}$, $x_k \in \mathbb{R}^m$ - set of data vectors
- $\mathbb{H}(\cdot)$ - LSH Function with bucket radius=r, dim=d
- $\mathbb{P} = \{p_1, ...p_n\}$ - set of n, $m \times d$ matrices w/ *JL* property
- $C : h \to 0$ - empty map of hashes to counts
- $M : h \to \langle 0, \ldots, 0 \rangle$ - empty map of hashes to vectors
- clusterer(M, k) - a standard clustering algorithm

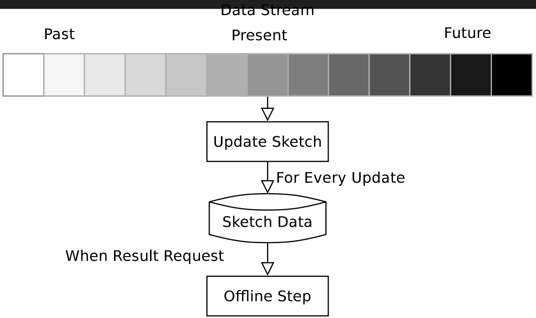
Phase 1: Counting Dense Regions

forall the $x_k \in X$ **do**
  forall the $p_i \in \mathbb{P}$ **do**
    $\tilde{x}_k \leftarrow \sqrt{\frac{m}{d}} p_i^{\intercal} x_k$
    $t = \mathbb{H}(\tilde{x}_k)$
    $C[h] += 1$
  end
end
C.sort()
Result: $C[0 : k * P.length])$

Phase 2: Cluster Assignment

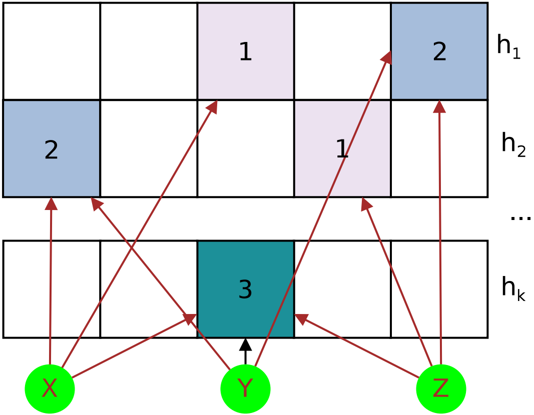
```
forall the $x_k \in X$ do
    forall the $p_i \in \mathbb{P}$ do
        $\tilde{x}_k \leftarrow \sqrt{\frac{m}{d}} p_i^\intercal x_k$
        $h = \mathbb{H}(\tilde{x}_k)$
        if $h \cap C.keys = \emptyset$ then
            $\Delta = M[h] - x_k M[h] = M[h] + \Delta / C[h]$
        end
    end
end
Result: clusterer($M, k$) ;                // merge centroids
```

Streaming Model

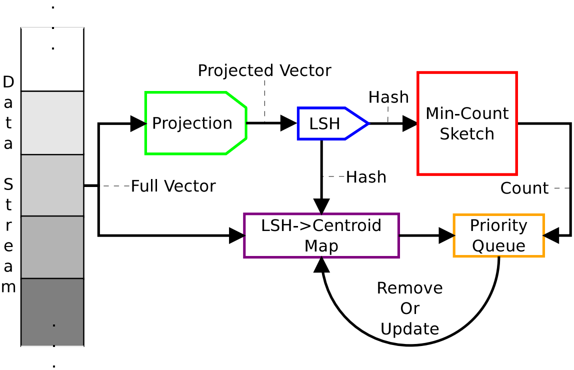


- Data arrives as a stream
- No random access to all data vectors
- data stream is unbounded
  - algorithm must be sub-quadratic
- Strict Memory bound

CountMin Sketch Structure



Streaming RPHash Diagram



Streaming Algorithm

Data

- k - number of clusters
- $x \in \mathbb{R}^m$ - data vector from stream
- $\mathbb{H}(\cdot)$ - LSH Function radius=r, dim=d
- $\mathbb{P} = m \times d$ matrix w/ JL property
- M- lsh_key $\rightarrow$ centroid map
- C- cm-sketch data structure
- T- CM-Sketch based ϵk bounded priority queue

Add Next Vector

forall the $x \in X$ **do**

$\tilde{x} := \sqrt{\frac{m}{d}} p^{\intercal} x$

$t := \mathbb{H}(\tilde{x})$ C.add(t)

if $t \in M.keys$ **then**

$M[t].wadd(x)$

else

$M[t]$ =new centroid(x,C.count(t))

T.insert($M[t]$)

end

M.remove(T.pop())

end

First Round Experiments

- Define evaluation metrics
- Review Datasets
  - Real World
  - Synthetic
- Optimize RPHash Configuration Parameters
- Algorithms for Comparison
- Standard RPHash Results
- Streaming Results

Experiment Metrics

- Internal performance metrics - unlabeled metrics
  - WCSSE - within cluster sum of squares
- External performance metrics - require ground truth labels
  - ARI - Adjusted Rand Index
  - Cluster Purity
- Runtime
- Memory Consumption

Datasets

- Synthetic data generators in R and Java
  - Uniformly Distributed Models
  - Uneven sized clusters
  - Gaussian distributed vectors
  - Sparse
  - Noise injected

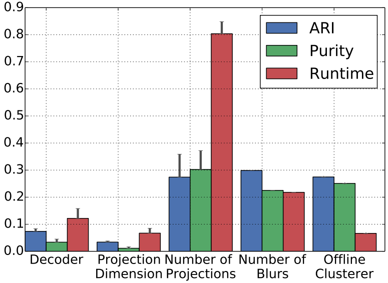
Data Set	**Num of Clusters**	**Num of Features**	**Num of Vectors**	**Type of Data**
Arrhythmia [**?**]	16	279	452	Real
CNAE-9 [**?**]	9	856	1080	Binary
Cora [**?**]	7	1433	2708	Binary
Gisette [**?**]	2	5000	7000	Real
Human Activity [**?**]	6	561	10299	Real
UJIIndoorLoc [**?**]	3	520	21000	Real
WebKB [**?**]	5	1703	265	Binary

RPHash Parameter Study

6400 Configurations!

***LSH* Algorithm**	**Projected Dimension(s)**	**# Projections**	**# Blurrings**	**Offline Clustering**
E_8	8	1, 2, 3, ..., 8	1, 2, 3, 4	*k*-means Single Linkage Complete Linkage Average Linkage
Multi-E_8	8, 16, 24, 32			
Leech	24			
Multi-Leech	24, 48, 72, 96			
Lévy p-stable	8, 16, 24, ..., 80			
Cauchy p-stable				
Gaussian p-stable				
Spherical				

Best Configurations



Data Set	Configuration	RPHash				k-means			
		ARI	Purity	Runtime	Memory	ARI	Purity	Runtime	Memory
Arrhythmia	Multi-E_8/24/1/2/Complete Linkage	0.2710	0.5885	0.1277	0.1420	0.0811	0.6069	0.5287	16.4333
CNAE-9	Multi-Leech/72/5/2/k-means	0.3424	0.5707	0.7917	0.8312	0.2798	0.5312	2.7120	165.1167
Cora	Multi-Leech/96/3/3/Complete Linkage	0.1065	0.3988	2.6022	0.6660	0.1158	0.4271	52.410	227.9833
Gisette	Spherical/16/2/3/k-means	0.5675	0.6218	1.7769	0.4530	0.4610	0.6002	24.746	1485.0667
UJIIndoorLoc	Spherical/8/8/2/k-means	0.6608	0.8268	1.3840	0.2780	0.6954	0.7750	23.821	2850.6500
WebKB	Spherical/16/5/1/k-means	0.4210	0.7396	0.1117	0.9015	0.4403	0.7528	0.8087	50.1500

Comparison algorithms

Static Clustering:

- *k*-**Means:** [?].
- **Agglomerative Hierarchical clustering:** Single, Complete, Average and Ward's method.
- **Self-organizing Tree Algorithm (SOTA):** [?].
- *k*-**Means++** [?]

Streaming Clustering

- **Streaming** *k*-**Means:** [?]
- **Damped Sliding Window:** [?]
- **DStream:** [?]
- **Biased Reservoir Sampling:** [?]

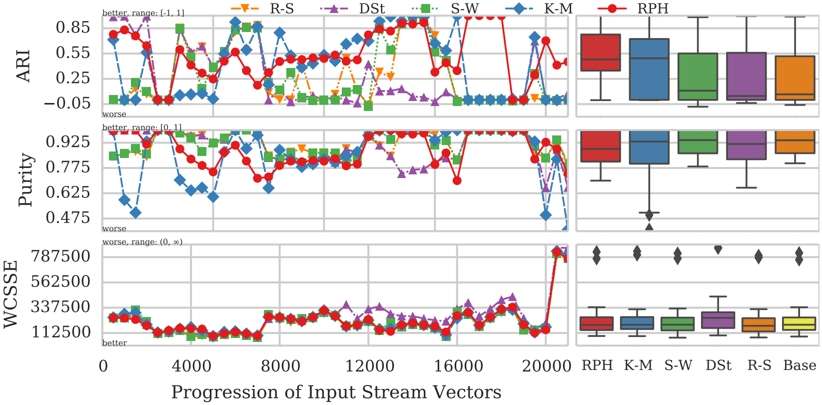
Experiments: Real Data

Data Set	Measures	RPHash	*k*-means [?]	Single Linkage	Complete Linkage	Average Linkage	Ward's Method [?]	SOTA [?]
Arrhythmia	ARI	0.0697	0.0811	0.0461	0.0963	0.0546	0.0889	0.0981
	Purity	0.6058	0.6069	0.5730	0.5885	0.5752	0.5951	0.6062
	Runtime	0.2709	0.5287	0.1680	0.1640	0.1680	0.1680	3.4440
	Memory	0.7070	16.4333	3.4000	3.4000	3.4000	3.4000	21.3000
CNAE-9	ARI	0.2788	0.2798	0.0000	0.0000	0.0000	0.3547	0.1730
	Purity	0.4873	0.5312	0.1185	0.1204	0.1185	0.5722	0.3657
	Runtime	0.3932	2.7120	4.3360	4.3400	4.3400	4.3440	3.7080
	Memory	1.1370	165.1167	24.2000	24.2000	24.2000	24.1000	134.2000
Cora	ARI	0.0915	0.1158	0.0001	0.0120	0.0002	0.0930	0.0647
	Purity	0.3858	0.4271	0.3039	0.3335	0.3039	0.4597	0.3342
	Runtime	0.8290	52.4100	71.9200	71.9600	71.9400	71.9720	11.7120
	Memory	1.4590	227.9833	100.7000	100.7000	100.7000	100.7000	265.4000
Gisette	ARI	0.1282	0.0615	0.0000	0.0000	0.0000	0.0018	0.1147
	Purity	0.6720	0.6241	0.5001	0.5003	0.5001	0.5216	0.6694
	Runtime	2.7363	423.7660	2280.4320	2280.4640	2280.0800	2280.5480	46.8120
	Memory	1.4300	2138.3833	829.3000	829.3000	829.3000	829.3000	2097.5000
HAR	ARI	0.3348	0.4610	0.0000	0.3270	0.3321	0.4909	0.3143
	Purity	0.4631	0.6002	0.1890	0.3770	0.3588	0.6597	0.3966
	Runtime	1.8774	24.7460	413.8800	414.3320	414.0960	414.4480	14.2440
	Memory	0.5157	1485.0667	1259.0000	1214.9000	1214.8000	1214.9000	946.2000
UJIIndoorLoc	ARI	0.5043	0.6954	0.0001	0.0001	0.0001	0.6021	0.3351
	Purity	0.7105	0.7750	0.4635	0.4635	0.4635	0.7732	0.6918
	Runtime	2.6363	23.8213	1093.9440	1094.7000	1094.8200	1095.5360	16.1880
	Memory	0.2460	2850.6500	5132.4000	5049.0000	5049.0000	5049.0000	2227.0000
WebKB	ARI	0.3205	0.4403	0.0066	0.0404	0.0066	0.3276	0.3906
	Purity	0.7063	0.7528	0.4755	0.5283	0.4755	0.7094	0.7019
	Runtime	0.1648	0.8087	0.3760	0.3760	0.3760	0.3760	2.5400
	Memory	1.2000	50.1500	6.3000	6.4000	6.3000	6.3000	44.1000

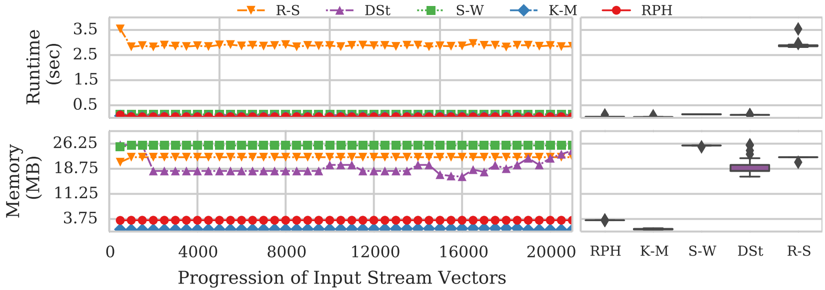
Streaming RPHash

- Test on UJII Indoor Localization
- High Dimensional (d=561)
- Many observation (21000)

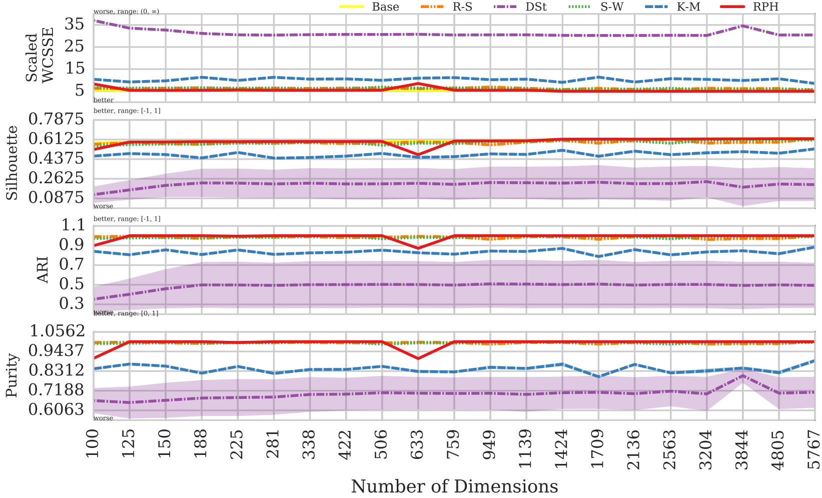
Streaming RPHash



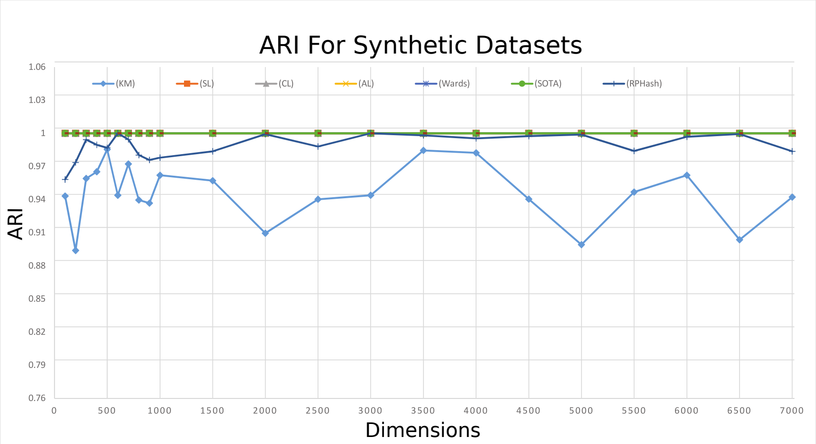
Streaming RPHash



Streaming RPHash



Performance Instability



- RPHash and *k*-Means Results are unstable
- Testing shows that RP and Counting are not the problem
- Problem must lie in the LSH Function

Better LSH

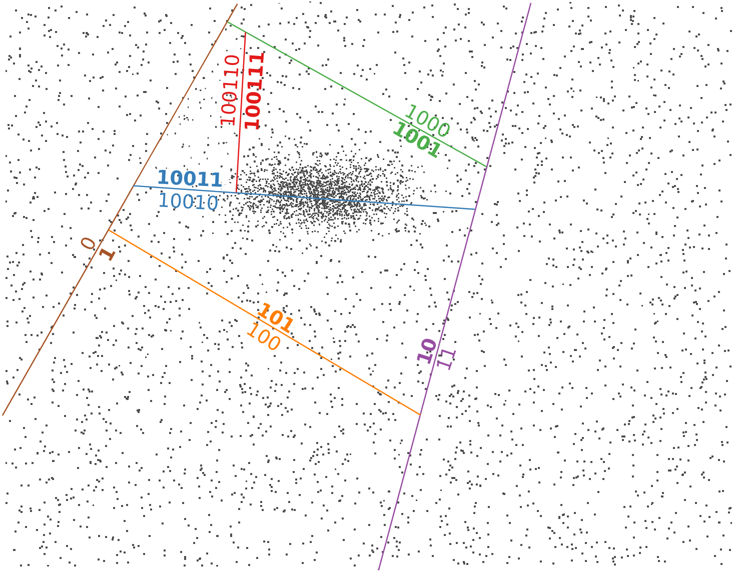
Fix The LSH Functions

- LSH functions are good when data is uniformly distributed
- clusterable data by definition is not uniformly distributed
- use a set of nested hash functions to adapt to data

Definition (LSH Composability)

An LSH function $\mathbb{H}^n(x)$ that maps $x \in \mathbb{R}^n \to \mathbb{Z}_2^n$, is composable if there is a related function $\mathbb{H}^{n-1}(x_{n-1})$ that maps $x_{n-1} \in \mathbb{R}^{n-1} \to \mathbb{Z}_2^{n-1}$ where $\mathbb{H}^{n-1}(x_{n-1}) = \left(\mathbb{H}^n(x) + 1\right) \bigcup \left(\mathbb{H}^n(x) + 0\right)$ for all $x_n \in \mathbb{R}^n$

Adaptive LSH



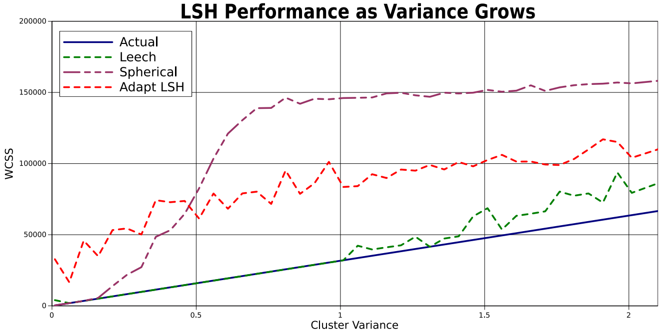
Adaptive LSH Algorithm

Definition (Sign based Projected LSH)

$$H(X) = \sum sign(P(X))2^n$$

$i = 1$
$ct, ct\_prev = C\left(\mathbb{H}^{i+1}(x)\right), C\left(\mathbb{H}^{i}(x)\right)$
while $i < n$ ***and*** $2ct > ct\_prev$ **do**
    $ct\_prev, i = ct, i + 1$
    $ct = C\left(\mathbb{H}^{i}(x)\right)$
end
return $\mathbb{H}^{i}(x)$

Adaptive LSH Performance

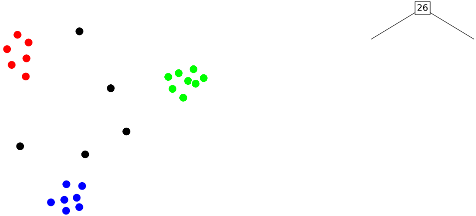


Worse Than Leech and Spherical

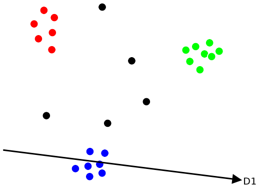
Evaluate the Neighbors

- Composable hashes let us investigate neighbors
- Use neighbor and parent relationships to decide when cuts are useful
- Generates a Tree
- Tree Based Clustering

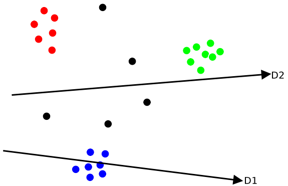
Tree Walk RPHash

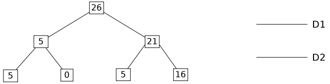


Tree Walk RPHash

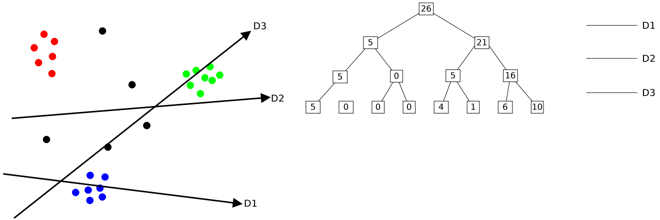


Tree Walk RPHash

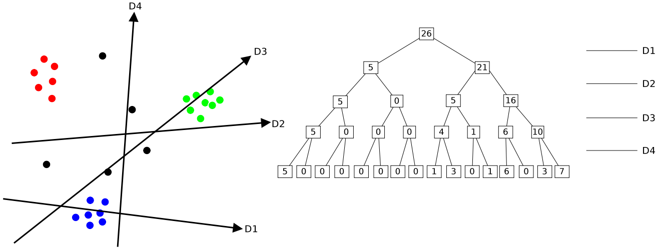




Tree Walk RPHash



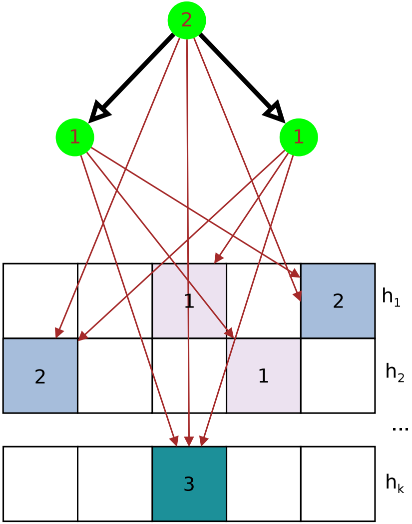
Tree Walk RPHash



Too Big

- The tree is exponential in depth
- $\theta(2^d * m)$ - storage complexity
- Just need to search, most nodes have low support
- Count-Min sketch

Count-Min Tree



Generate Tree

UNIVERSITY OF Cincinnati

forall the $x \in X$ **do**

$\tilde{x} = \sqrt{\frac{m}{d}} p^{\intercal} x$ $h := \mathbb{H}(\tilde{x})$

while $h > 0$ **do**

$h = h \gg 1$

$x' = C[h] + x$

$C.\text{add}(h, x')$

end

end

k -number of clusters

$X = \{x_1, ..., x_n\}$, $x_i \in \mathbb{R}^m$

C- cm-sketch,

counts$\rightarrow$vector

$\gg$ - bit shift

$\mathbb{H}(\cdot)$ - LSH Function

$\mathbb{P} = \{p_1, ...p_n\}$ - Projectors

$+$-weighted addition

CLTree for High Dimensional

- TWRP was developed independently, but similar
- Concern of CLTree is intersecting clusters
- We ignore this concern for high dimensional data and proof the following theorem

Theorem (Hyper-rectangle Splitting)

.

The probability of splitting a hyper-rectangular region into two equal mass clusters where subsequent dimensional cuts are always of the smaller region is 0 as the dimensionality grows to infinity.

$$\lim_{d \to \infty} \frac{Vol(R) - Vol_{removed}(R)}{Vol(R)} = 0, \textit{ R is a hyper-rectangle in } \mathbb{R}^d$$

Evaluate Tree

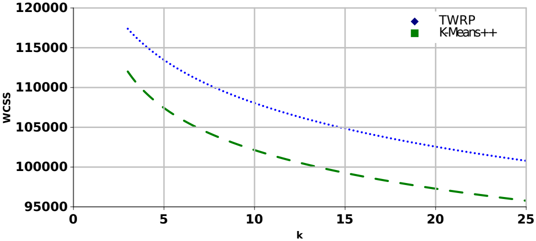
UNIVERSITY OF Cincinnati

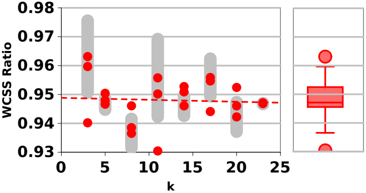
forall the $H \in sort(C.ids)$ **do**
  if $2C[H] < C[H \gg 1]$ **then**
    $C[H \gg 1] = 0$
  end
end
$L = []$
forall the $h \in sort(C.counts)$ **do**
  $L \leftarrow \text{medoid}(C[H])$
end
return L

Further Experiments

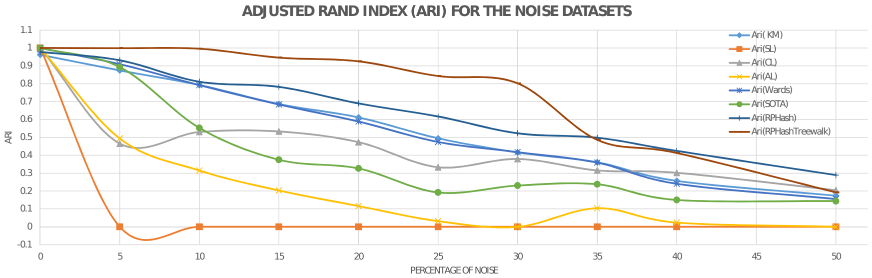
- Compare Scalability of algorithms
- Parallel Speedup Comparison
- Security Evaluation on Real Data

Word2Vec TWRP



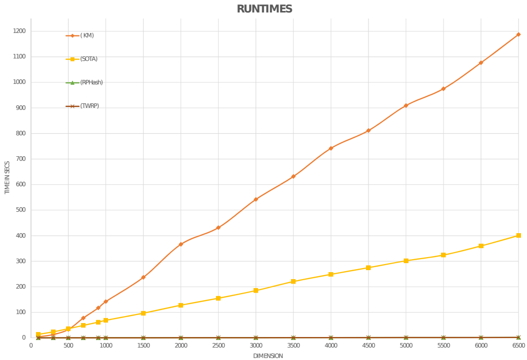


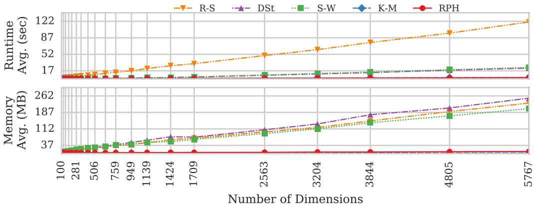
TWRP Noise



ADJUSTED RAND INDEX (ARI) FOR THE NOISE DATASETS
ARI
PERCENTAGE OF NOISE
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
5
10
15
20
25
30
35
40
45
50
Ari(KM)
Ari(SL)
Ari(CL)
Ari(AL)
Ari(Wards)
Ari(SOTA)
Ari(RPHash)
Ari(RPHashTreewalk)

Scalability

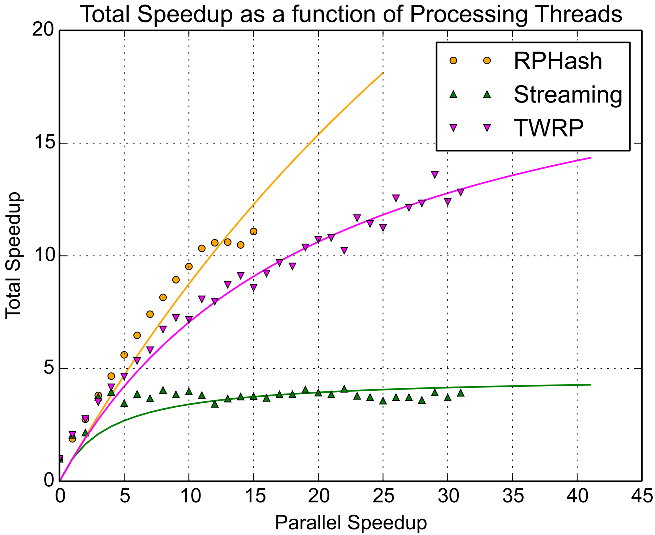




Theoretical Complexity

LSH **Algorithm**	**Time Complexity**	**Space Complexity**
RPHash	$\Theta(nm)$	$\Theta(nm)$
Streaming RPHash	$\Theta(nm)$	$\Theta(m \log\log(n))$
TWRP	$\Theta(nm\log^2(n/\varepsilon))$	$\Theta(m\frac{e}{\varepsilon} ln(\frac{nlog(d)}{(\sqrt{\delta})}))$

Parallelism



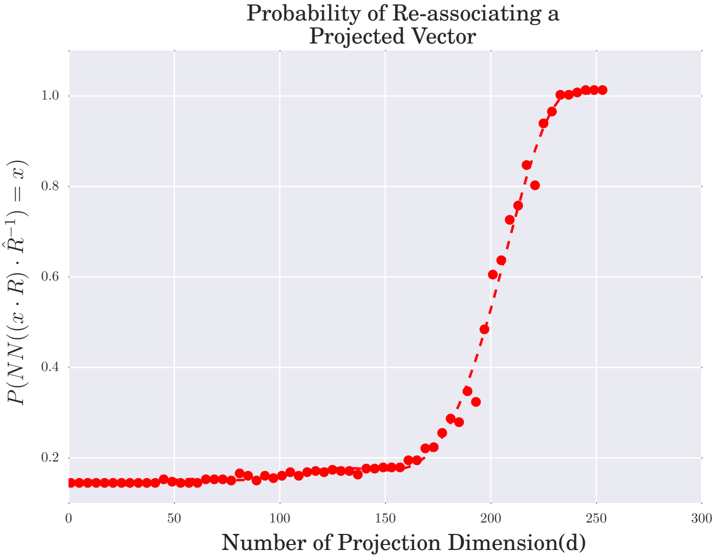
Security Assessment

- *RPHash* increases granularity by projection
- Similar to the *l*-diversity metric
- Evaluate random projection against data recoverability

$$u = \sqrt{\frac{n}{k}} R^T_{d \to s} v, \; v' = \sqrt{\frac{k}{n}} u^T R^{-1}_{s \to d}$$

- *R* is non-invertible, best we can do is the Moore-Penrose pseudo-inverse.

Security Assessment (conti.)



Conclusions

UNIVERSITY OF Cincinnati

- Empirically we show that TWRP algorithm and both streaming and standard RPHash are comparable to other clustering methods
- our hypothesis that approximate clustering vs local minima clustering holds for many real world and synthetic datasets
- RPHash has linear complexity, and memory bound and in the streaming case sub-linear memory bound both in theoretically and shown in experiments.

Future Work

UNIVERSITY OF Cincinnati

- Count-Min Cut Tree is interesting for approximate data analysis
- Topological Data Analysis could potentially use RPHash for micro-cluster identification to accelerate it
- Could accelerate hashing with GPUs

Questions??

References I

C. C. Aggarwal.
On biased reservoir sampling in the presence of stream evolution.
In *Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06*, pages 607–618, Seoul, Korea, 2006.

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L.
A public domain dataset for human activity recognition using smartphones, Apr. 2013.

D. Arthur and S. Vassilvitskii.
K-means++: The advantages of careful seeding.
In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages

1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

V. Braverman, A. Meyerson, R. Ostrovsky, A. Roytman, M. Shindler, and B. Tagiku.
Streaming k-means on well-clusterable data.
In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 26–40. SIAM, 2011.

P. M. Ciarelli and E. Oliveira.
UCI machine learning repository.

H. A. Guvenir.
UCI machine learning repository.

I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror.
Result analysis of the NIPS 2003 feature selection challenge, 2004.

J. A. Hartigan and M. A. Wong.
A k-means clustering algorithm.
JSTOR: Applied Statistics, 28(1):100–108, 1979.

J. Herrero, A. Valencia, and J. Dopazo.
A hierarchical unsupervised growing neural network for clustering gene expression patterns, 2001.

B. Liu, Y. Xia, and P. S. Yu.
Clustering through decision tree construction.
In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 20–29, New York, NY, USA, 2000. ACM.

References IV

F. Murtagh and P. Legendre.
Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?
J. Classif., 31(3):274–295, Oct. 2014.

H. Samet.
Foundations of Multidimensional and Metric Data Structures.
Morgan Kaufmann, 2006.

P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad.
Collective classification in network data.
AI Magazine, 29(3):93–106, 2008.

References V

J. Torres-Sospedra, R. Montoliu, A. Martinez-Uso, T. J. Arnau, J. P. Avariento, M. Benedito-Bordonau, and J. Huerta.
Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems, 2014.

L. Tu and Y. Chen.
Stream data clustering based on grid density and attraction.
ACM Trans. Knowl. Discov. Data, 3(3):12:1–12:27, July 2009.

References VI

Y. Zhu and D. Shasha.
Statstream: Statistical monitoring of thousands of data streams in real time.
In *Proc of the 28th Int Conf on Very Large Data Bases*, pages 358–369, 2002.