Random Projection Hashing for Scalable Big Data Clustering

Lee Carraher

February 4, 2014

Abstract

1 Project Overview

RPHASH is a novel method proposed for large scale distributed data clustering using locality sensitive hash functions and random projection. Clustering is an inherent problem in data analysis that faces a variety of issues when applied to very large, distributed datasets. The system designed herein will focus on data security, algorithmic and communication scalability. The system will be compatible with commercial, cloud-based, distributed computing resources(e.g. Amazon EC2, Google Cloud, and similar offerings from IBM and Intel) as well as private distributed processing systems. The proposed system will provide a straightforward interface running on the popular MapReduce (*MRv2*) [1] parallel processing system, to allow for arbitrary resource scalability. The clustering system proposed will also have wide applicability to a variety of standard clustering applications while focusing on a subset of clustering problems in the biological data analysis space. Furthermore, due to new requirements involving the handling of health care data, as well as the inherent privacy concerns of using cloud based services, data security is primary to RPHash'es interprocess communication architecture.

1.1 RPHash: A Secure, Scalable Clustering Algorithm

The RPHash algorithm uses various recent techniques in data mining along with a new approach toward achieving algorthmic scalability. A large body of research has been applied to this field and in the past has focused on shared memory architectures with heavy data bandwidth requirements. Recent architectures however, promoted by the commercial successes of Big Data and cloud based on commodity hardware, have driven down system bandwidth, shifting related requirements for data clustering algorithms. RPHash is an algorithm designed with the priority of minimizing interprocess communication requirements, in exchange for some redundant computation. As with many parallel computations, and often computing in general, memory access and synchronization bottlenecks often dominate the computational complexity of an algorithm. For this reason, we propose redundancy of computation as an acceptable trade-off, while hopefully achieving similar results of other communication intensive clustering algorithms.

Our clustering system is intended for use by a variety of researchers in fields beyond computing, and thus must require a minimal learning curve. One such way of providing this, is by removing the technical hurdle of building and maintaining large scale computing systems, and instead rely on commercially provided cloud computing resources. Furthermore, as the distributed processing architecture is a commercially provided system, the companies commercial ambitions will promote ongoing access to state-of-the-art computing resources. Cloud computing provides a pay-per-use computing resources that scale with the computational requirements without burdening researchers or their IT providers.

As RPHash is intended for cloud deployment, data security should be a primary concern. Privacy concerns over attacks on de-identification, or de-anonymization [2] [3] of thought to be anonymous data, such as whole genome sequence data [4], has prompted a presidential commission on the subject [5].

The results of which add new restrictions to the way researchers acquire and process data. While attempting to mitigate communication restrictions, RPHash intrinsically provides some security in the data it chooses to exchange. Namely, the randomly projected hash IDs, and the aggregation of only the k-largest cardinality vectors sets. Non-distributed data clustering requires the entire dataset to reside on the processing system, while distributed methods often require communication of full data records between nodes. In our approach, nearly-orthogonal random projection is a destructive operation, that provides privacy of the data being processed and communicated. While centroid communication is by definition only transmitting the aggregate of data records, and in the case of patient data, is not any single individual's information.

1.2 Proposal Objectives

The overall goal of the proposal is to create a secure, and scalable cloud based clustering system of high dimensional data such as genomic data. The following research objectives will be addressed specifically:

1. **Develop the sequential RPHash Algorithm:** The first step in developing our parallel system is in developing the initial sequential version of the RPHash algorithm. From previous work, the difficult implementation of decoders and random projection methods have already been developed [6] , the overall algorithm needs further implementation improvements to accurately match the sequential version of the proposed parallel algorithm.

2. **Develop A distributed version of the proposed RPHash Algorithm:** As our current sequential algorithm for random projection hashing has shown both accuracy and scalability promise, our next step is to extend the system to a parallel architecture for which it was designed for. The parallel extension is a straightforward application of the parallel prefix sum algorithm of Ladner [7] providing work efficient parallel speed up.

3. **Deploy RPHash on Map Reduce(MRv2) Hadoop framework for parallel computation:** Building the parallel RPHash for the standard open source map reduce implementation will allow us to test the scalability performance using both internal and publically available cloud resources. In addition, due to Hadoop and public cloud computing resources availability, this goal will serve as a roadmap for other researchers use of the developed system.

4. **Compare the Accuracy, Speedup, and Scalability of RPHash:** The availability of state of the art data clustering algorithms for Hadoop, provided by the Mahout Data Mining library, will allow us to compare the accuracy, performance and scalability of our system against a variety of optimized alternatives. Comparisons will be made based on overall clustering accuracy, such as precision recall, scaled speed, and overall problem size scalability.

5. **Show RPHashes innate resistance to de-anonymization attacks:** Though proving security without rigorous proof is an open problem, we hope to show empirically the resistance of RPHash to various de-anonymization attacks and data scraping. This rudimentary analysis, wil focus on understanding the amount of destructiveness the random projection method has on data and its resistance to least squares inversions.

1.3 Impact

In this proposal we will develop a a novel algorithm for scalable data clustering in the cloud. The algorithm developed combines a variety of approximate and randomized methods for providing solutions to real world data problems. Our focus on a non-deterministic clustering algorithm is somewhat uncommon in computing, but coincides well with the ill-posed problem of clustering. This follows naturally from a similar inversion regarding clustering and computing, in which real world problems tend to converge faster than theoretically posed problems. Furthermore, the addition of noise to theoretically posed problems often yield more favorable results than the theoretical problem alone.

Clustering is one of the most commonly used methods for the analysis of unlabeled data. Fields in health and biology are benefited by data clustering scalability. Such fields as Micro Array clustering,

Protein-Protein interaction clustering, medical resource decision making, medical image processing, and clustering of epidemiological events all serve to benefit from larger dataset sizes.
In this proposal we plan to create a highly scalable, cloud based data clustering system for health care providers and researchers. This system will allow researchers to scale their clustering problems without the need for specialized equipment or computing resources. Our cloud processing solution will allow researchers to arbitrarily scale their processing needs using virtually limitless commercial processing resources.

2 Background

Data clustering is a useful tool in pilot data and knowledge engineering. It's uses range from practical data localization problems, to helping shape our understanding of complex biological, social, and physical structures. Though the problem itself has been shown to be NP-Hard [8] [9], many algorithms exist that usually terminate in polynomial time with good results.

2.1 Big Data Clustering

In light of the "Big Data" revolution, many machine learning algorithms have been modified to work on very large, distributed datasets. The goal of clustering is the same, however memory and network constraints tend to dominate the running time requirements over algorithmic complexity alone. RPHash is a clustering algorithm designed from the ground up to deal with these problems head on. Instead of modifying an existing algorithm to perform better across networks and distributed systems, RPHash trades redundant computation for lower memory and network bandwidth requirements through the use of random projections, and universally deterministic hashes.

2.2 Random Projection Methods

The resurgence of the random projection method of Johnson and Lindenstrauss was reinvigorated with the work of Achlioptas on Database Friendly Projections [10], that provided good subspace embeddings requiring minimal computation cost.

Theorem 2.1 (JL - Lemma)**.**

$$(1-\epsilon)\|u-u'\|^2 \leq \|f(u)-f(u')\|^2 \leq (1+\epsilon)\|u-u'\|^2$$

Johnson-Lindenstrauss lemma gives a tight bound for discretion of n vectors subjected to random projections in $\mathbb{R}^d$, $\Theta(\frac{log(n)}{\varepsilon^2 log(1/\varepsilon)})$. Vampala gives a relaxation of the JL-bound of $d = \Omega(n)$ [11], with a scaling factor of $\frac{1}{\sqrt{n}}$ to preserve approximate distances between projected vectors.

An additional benefit of Random Projection for mean clustering, is that randomly projected asymmetric clusters tend to be spherical in lower dimensional subspace representations [12].

2.3 Discrete Subspace Quantizers

From the continuous spaces of random projections we switch to discrete space partitions lattices.

Definition 2.2 (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 $Pr_{\mathbb{H}}[h(u) = h(v)] \leq p_2$*

2.4 Issues with random projection and high dimensions

The **Curse of Dimensionality** Consider the distance between any two points x and y in $\mathbb{R}^d$. under Euclidean distance: $dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$. Now if we consider the vector as being composed of values that have a uniformly distribution, according to Ullman [13] the range of distances are all very near the average distance between points. Given this, it becomes difficult to differentiate between near and far points, or in other words, the distance function looses its usefulness for high dimensions [14]. The usefulness of this principle metric in clustering suggests that approximate and exact algorithms will have an intersection at some number of dimensions. Though this intersection is not guaranteed to be at a point of favorable clustering accuracy, Vempala suggests random projection actually may solve some of these issues [11] [12].

The **Occultation Problem** arises when two discrete clusters in d-dimensional space project in to two non-discrete clusters in a lower dimensional space. Our use of 24 dimensions helps mitigate this problem. The probability occultation for d-dimensional space to 1-dimensional space is given in [15]. Repeated projections result in a probability of occultation for every projection converging to 0 as the number of projections increases. In the case of 24 dimensional projection, we achieve a slightly faster convergence as the 24-dimension random subspace's vectors are nearly orthogonal for Gaussian distribution [11]. Furthermore the convergence from Urruty: $\underset{d\to\infty}{lim} 1 - \left(\frac{2(r_1+r_2)}{\pi\|d-c\|}\right)^d$ is exponential in d.

2.5 Mapreduce, Hadoop, and Cloud

Mapreduce is a framework for data parallel processing using two functional programming mainstays, the map and reduce function, In addition to providing a standard parallel algorithmic structure, mapreduce also performs many of the standard resource management tasks required for parallel processing.

Hadoop is a freely distributed, open source implementation of the mapreduce framework. Currently in its second version utilizing the new Yarn resource manager, Hadoop will provide the networking tasks required for RPHash.

Cloud is a somewhat nebulous term as it is often used to describe anything involving web services. In regards to this proposal, cloud will always refer to externally provided computing resources, and in particular Amazon's implementation of Hadoop on their EC2 web service.

3 Proposed Research

Clustering algorithms offer insight in a multitude of data analysis problems. Clustering algorithm are often limited in their scalability due to communication requirement bottlenecks. A variety of methods have been proposed to combat this issue with varying degrees of success. In general, these solutions tend to be parallel patches and modification of existing clustering algorithms. *RPHash* is a clustering algorithm designed uniquely for low interprocess communication distributed architectures. Random projection clustering has been previously explored in [15], [16], [17], [18], on single processing architectures.

Despite theoretical results showing that k-means has an exponential worse case complexity [19], many real world problems tend to fair much better under k-means and other similar algorithms. For this reason, clustering massive datasets, though suffering from unbounded complexity guarantees, is still a very active area in computing research. Approximate and randomized methods are common tools for overcoming complexity growth. In our algorithm, called Random Projection Hash (*RPHash*) algorithm, we utilize both approximate and randomized techniques to provide a scalable, approximate parallel system for massive dataset clustering. RP-Hash performs multi-probe, random projection of high dimensional vectors to be hashed into the unique subset boundaries of the Leech Lattice(Λ_{24}).

3.1 Objective 1: Develop the Sequential RPHash Algorithm

3.1.1 sequential algorithm

log-near searches around buckets [20]
explanation and complexity analysis

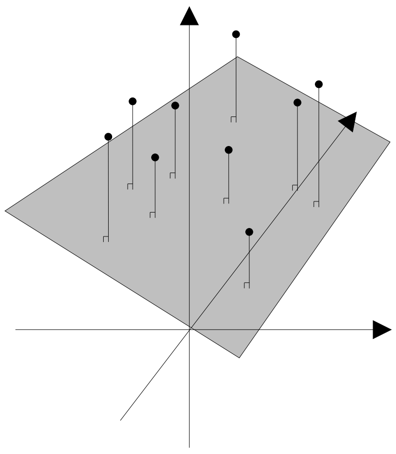
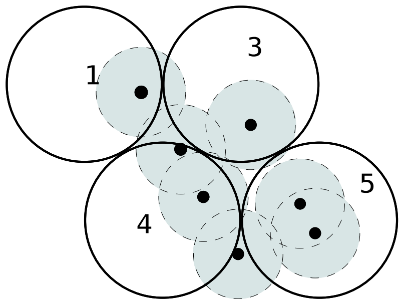


Figure 1: Random projection on a plane



Multiple Projection Region Assignments and Cardinalities

sequentially its a dog

An outline of the steps in *RPHash* is given below, however we would also like to highlight some aspects of its function in regards to randomness and approximation. One way in which the *RPHash* algorithm achieves scalability is through the generative nature of its region assignment. Clustering region assignments are performed by decoding vector points into partitions of the Leech Lattice. The Leech Lattice is a unique lattice that provides optimal sphere packing density among 24 dimensional regular lattices [21]. Though optimal, due to the curse of dimensionality, the overall density is somewhat sparse, requiring that the algorithm apply shifts and rotations to the lattice to fully cover the $\mathbb{R}^{24}$ subspace. Furthermore, in general vectors will be greater than 24 dimensions. We cite the Johnson-Lindenstrauss (*JL*) lemma to provide a solution to this problem (Figure 1). *JL* states that for an arbitrary set of n points in m dimensional space, a projection exists onto a d-dimensional subspace such that all points are linearly separable with ϵ-distortion following $d \propto \Omega(\frac{log(n)}{\epsilon^2 log 1/\epsilon})$. Though many options for projections exists, a simple and sufficient method for high dimensions is to form the projection matrix $r_{ij} \in \mathbf{R}$ is $m \times d$ as follows.

$$r_{ij} = \begin{cases} +1, & \text{with probability } \frac{1}{6} \\ 0, & \text{with probability } \frac{2}{3} \\ -1, & \text{with probability } \frac{1}{6} \end{cases}$$

Overview of the sequential algorithm The basic intuition of *RPHash* is to combine random projection with discrete space quantization, and regard regions of the k-highest density as centroid candidates. According to *JL* lemma, the sub-projections will conserve the pairwise distances in the projected space for points with ϵ-distortion. In addition to compressing a dataset to a computationally more feasible subspace, random projection can also make *eccentric* cluster more spherical [22] [11].
The next piece of *RPHash* are discrete space quantizers. For our implementation of *RPHash* we will rely on the Leech lattice as our region quantizer. The Leech lattice unique lattice in 24 dimensions that is the

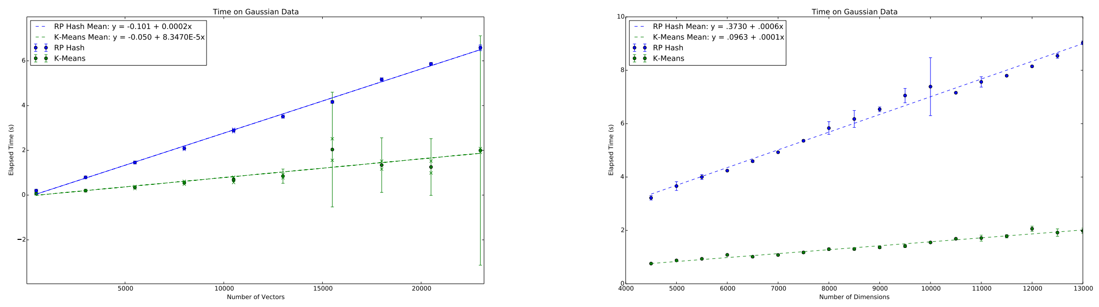


Figure 2: Computation Time for K-Means(green) and RPHash(blue) varying Vectors and Dimensions

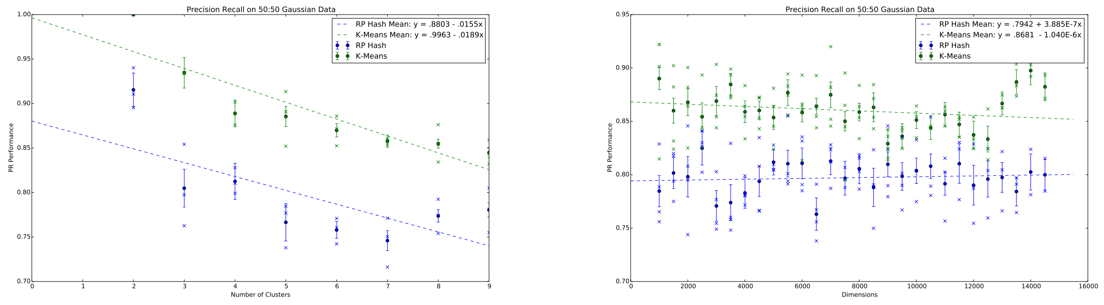


Figure 3: Precision-Recall for K-Means(green) and RPHash(blue) varying Vectors and Clusters

densest lattice packing of hyperspheres in 24 dimensional space. The Leech lattice is an optimal regular lattice space quantizer. The 24 dimensional subspace partitioned by the Leech Lattice is small enough to exhibit the spherical clustering benefit of random projection. Low distortion random embeddings are also feasible for very large datasets ($n = \Theta(c^{24})$) objects while avoiding the occultation phenomena [15].

Furthermore, the decoding of the Leech lattice is a well studied subject, with a constant worse case decoding complexity of 331 operations [23].
Space quantizers have hard margin boundaries and will only correctly decode points that are within the error correcting radius of its partitions. This is an an issue found in approximate nearest neighbor search [20] [24], and we choose to overcome it in a similar way as in Panigrahy [20] by performing multiple random projections of a vector then applying the appropriate locality sensitive to provide a set of hash IDs. Using many random projections of a vector allows the high dimensional vector to be represented as 'fuzzy' regions probabilistically dependent on the higher dimensional counterpart. From Panigrahy [20], we are given a bound of $(\Theta(log(n)))$ for the required number of random projection probes to achieve hash collisions for a bounded radius, r -near vectors. Figure **??** shows an example of this process as a set of probabilistic collision regions. Random projection probing adds a $\Theta(log(n))$ -complexity coefficient to the clustering algorithm but is essential for compute node independence as will be addressed in the parallel extension of this algorithm. The intuition of this step is to find lattice hash IDs that on average generate more collisions, than others. The the top k cardinality set of lattice hash ID vector subsets represent regions of high density. The centroids are computed from the means of the non-overlapping subsets of vectors for each high cardinality lattice hash ID.

The algorithm used above has a natural extension to a parallel algorithm which we will describe in the following section.

3.2 Objective 2: Develop A distributed version of the proposed RPHash Algorithm

A key aspect of generative groups such as the IDs of lattice centers in the Leech Lattice, is that their values are universally computable. For distributed data systems this aspect provides complete compute node independence. Furthermore, r-near neighbor hash collisions for projected vectors do not need to share a particular basis. Instead they need only be randomly projected a bounded number of times according to results found in Panigrahy [20]. Using these tools, the *RPHash* algorithm is mostly naively parallel. The overall parallel portion of the algorithm is in fact no more complicated than a parallel sum algorithm, or reduce operation on Hadoop.

Algorithmic adaptations to parallel sum problem is yet another common tool for attaining parallel scalability and limiting complexity growth. Due to the work efficient parallel complexity of the reduce function, the overall complexity of RPHash is asymptotically no worse than its sequential complexity. This complexity, as previously stated is dominated by the number of required random projection hash probes for n vectors($\Theta(nlog(n))$). The data transfer requirement of the directly applied parallel sum for *RPHash* is on the order of the dimensionality of the vectors and the parameter k for each compute node. To minimize this, with no overall effect on the complexity order of Parallel *RPHash*, we propose a two step approach. The steps will be summarized here, as well as in standard psuedo-code algorithmic form (1 and 2 respectively).

Phase 1 of Parallel RPHash: Maximum Bucket Counting.
As *RPHash* is a distributed algorithm, we will assume the data vectors reside on D independent computing nodes, connected through any non-specific, Hadoop [25] compatible network hardware. The system architecture will utilize the Yarn *MR*v2 resource manager for task scheduling and network communication. With much of the parallel task scheduling details handled by Yarn *MR*v2, we focus our algorithm description on the per compute node task level. As the random projections can be performed independently, the compute node interdependency is very low. Due to the Leech lattice's set dimensionality, conversion between vectors of $\mathbb{R}^d$ and $\mathbb{R}^{24}$ will be performed by a random projection matrix of random variables following a nearly Gaussian distribution as in Bingham [12]. Therefor our first parallel task is to distribute a random projection matrix to each compute node. To further minimize communication load, we will note the fact that the random projection matrix, though, stated as random, is actually the result of a pseudo-random number generation sequence that is uniquely defined by a single seed number. A trade-off exists between the communication cost of distributing a large $\mathbb{P}_{m\to d}$ and the redundant calculation of such data per compute node. Parallel processing tuning is an effective solution to this problem, however in *RPHash* we choose the generative method, due to the constant scalability barrier that results from network bandwidth saturation and empirical results from Chowdhury [26] suggesting that the ratio of communication to computation increases with the number of MR nodes. With the initial data distribution requirements taken care of, we can return to focusing on per node computation. Each compute node processes its set of vectors independently; first projecting it to a random subspace, then decoding to its nearest Leech lattice centroid ID. The lattice region IDs are generative, and are thus identical across all compute nodes. In the 'bucket counting' phase, we are only concerned with the number of collisions each lattice ID receives. Once all data vectors have been processed and their collisions accumulated, we will use the standard parallel prefix sum algorithm of Ladner [7] to accumulate the lattice ID counts across all compute nodes. The set of lattice ID counts is sorted, and the $k\log(n)$-largest Leech lattice IDs are broadcast to all compute nodes for Phase 2 processing. $\Theta(n^\rho)$-time. For a Leech lattice hash function where $d \leq 24$, $\rho \approx 0.2671$ [24], while linear search requires $\Theta(dn)$-operations. This gives an internal clustering complexity of $\Theta(kn^{1+\rho})$ where k is the number of vectors per compute node.

Note: This may result in lost clusters, however preliminary results show our algorithm performs as good as or better than K-means for the same dataset, in dimensions greater than $d < 100$.

Show graph of precision recall here.

Phase 2 of Parallel RPHash: The two phased approach for clustering is motivated by the work of Panigrahy, Andoni, and Indyk [20] [24] in c-approximate nearest neighbor search(c-ANN). In the first phase, the search database of high cardinality lattice hashes is constructed/ While actual searching of the database is performed in the second phase. *RPHash* is a slight inversion of c-ANN, in the second phase, all vectors of the database are searched, against the small subset of high cardinality lattice hash ID sets. The set of lattice hash ID's corresponding to the $k * log(n)$ largest lattice hash ID subsets are broadcast to all compute nodes through standard MapReduce broadcast methods. In the case of

Algorithm 1 Phase1 RP-Hash Clustering

```
Require: $X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - data vectors
1: $D$ - set of available compute nodes
2: $\mathbb{H}$ - is a d dimensional LSH function
3: $\widetilde{X} \subseteq X$ - vectors per compute node
4: $\mathbb{P}_{m \to d}$ - Gaussian projection matrix
5: $C_s = \{\varnothing\}$ - set of bucket collision counts
6: for all $x_k \in \widetilde{X}$ do
7:     $\tilde{x_k} \leftarrow \sqrt{\frac{m}{d}}\mathbb{P}^{\intercal} x_k$
8:     $t = \mathbb{H}(\tilde{x_k})$
9:     $C_s[t] = C_s[t] + 1$
10: end for
11: sort($\{C_s, C_s.index\}$)
12: return$\{C_s, C_s.\text{index}\}[0 : k\log(n)]$
```

overlapping lattice hash set, clusters having similar centroids, an overlap factor of $log(n)$ is applied to the parameter k, so overlapping subsets can be merged during the gather phase with little effect on computational complexity.
The per compute node processing of Phase 2 requires that we perform the projection and hashing process from Phase 1 for $log(n)$ random projections. While processing the projection and hashes, we must also store the sums of the original vectors for each $klog(n)$ cluster IDs. After processing the $log(n)$-projections of n vectors, we use a similar parallel prefix scan method for vectors, to sum up all the vectors and counts. The means for each vector sum are then computed using the set of lattice ID collision totals.

Algorithm 2 Phase2 RP-Hash Clustering

```
Require: $X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - data vectors
1: $D$ - set of available compute nodes
2: $\{C_s, C_s.\text{index}\}$ - set of $klogn$ cluster IDs and counts
3: $\mathbb{H}$ - is a $d$-dimensional LSH function
4: $\widetilde{X} \subseteq X$ - vectors per compute node
5: $p_{m \to d} \in \mathbb{P}$ - Gaussian projection matrices
6: $C = \{\varnothing\}$ - set of centroids
7: for all $x_k \in \widetilde{X}$ do
8:     for all $p_{m \to d} \in \widetilde{\mathbb{P}}$ do
9:         $\tilde{x_k} \leftarrow \sqrt{\frac{m}{d}} p^{\intercal} x_k$
10:        $t = \mathbb{H}(\tilde{x_k})$
11:        if $t \in C_s.\text{index}$ then
12:            $C[t] = C[t] + x_k$
13:        end if
14:    end for
15: end for
16: return $C$
```

3.2.1 Drawbacks

For a fixed number of random projections the probablility of finding a goof projections converges exponentially to zero as the number of dimensions d increases. [16]

3.3 Objective 3: Deploy RPHash to EC2 Cloud's Hadoop Implementation

An important goal of RPHash and big data mining for the computing community in general is accessability to data scientists. RPHash is not focused on exotic, difficult to access systems, and instead is intended to be built for cloud processing platforms.

The first requirement of deploying our parallel algorithm on a cloud based platform is to create a simulated environment on a local system. Virtualization will be used to create this simulated environment, and scripts(Chef, Puppet) will be written to handle the automatic deployment of virtual machines running the Hadoop MRv2 framework. Automatic deployment scripts will help with speedup tests in objective 4 both locally and on the EC2 Cloud.

After a local deployment system is available, an Amazon EC2 cloud account will be created for further testing. The amazon account will be managed by the researchers using funds from this grant for Amazon's pay-per use computing services. Most EC2 testing will be performed around scalability testing on synthetic data. This requirement will avoid the expenses associated with Amazon's services, while also not violtating potential copyright and privacy issues that could arise from uploading to the cloud.

3.4 Objective 4: Demonstrate Accuracy, Speedup and Scalability of RPHash

As RPHash is a new method for clustering, its accuracy will have to be tested on a variety of types of data. Though specifically intended for high dimensional data, various synthetic, real, and combination datasets will be tested. Additionally RPHash will be compared with the current state-of-the-art parallel clustering algorithms.
Mahout is our primary resource for such algoriths, as they have been openly designed and extensively optimized for running on the Hadoop MRv2 engine. RPHash will be tested agains these systems in three areas of algorithmic interest.

- Accuracy - a 50:50 model training set will be created from the three types of datasets. Common accuracy measures will be taken such as the common Precision-Recall test, ROC test, and TF/FP tests. Though a variety of other tests exist for clustering accuracy, we mostly expect to approach the other algorithms of Mahout in accuracy, as our real goals are focused on the latter tests.
- Speedup - is simply a measure of how well we utilize the addition of processors. Unfortunately speedup is very commonly subject to Amhdals law, and the return on investment often falls of quickly. Though this is an almost unavoidable constraint on parallel algorithms, RPHash should perform very well as its sequential requirements are fairly small.
- Scalability - Another important test of any clustering algorithm is how well it scales with increasing dataset sizes. Though we have rough estimates from sequential tests and theoretical algorithmic design on how RPHash should scale, true empirical results will offer a more robust method of comparing with the algorithms of Mahout.

3.5 Objective 5: Data Security

Though certain privacies are provided by cloud service providers, the security itself is entirely dependent upon the often, private companies security practices. For certain problems, specifically those involving private individuals healthcare data, such privacy safegaurds must undergo rigorous auditing requirements. Such auditing is out of the scope of a private cloud companies business, making the move to cloud resources difficult for the health care community. Furthermore, recent United States government initiatives pushing for the large scale availability of data resources have made vast quantities of de-identified health information available to the public. These resources however have prompted advances in attacks on de-identification of whole genome sequence data. Such attacks have been used to to associate, thought to be, anonymous medical records with specific individuals [4]. Similar de-anonymization attacks [2] [3] along with a presidential commission (privacy and progress in WGS) have prompted a need for better data security of medical records data. Our algorithm provides an intrinsic solution to this problem in both the distribution of data among servers as well as during the communication steps required by the algorithm.

RPHash is a distributed algorithm designed to compute clusters on databases that could potentially span over the public internet. As such, data it uses to compute clusters can theoretically reside independently among different health care facilities with no requirement for any single location data storage

architecture. As a consequence of projecting the real data vectors to a random subspace via a near, but not completely orthogonal matrix, destructive data loss occurs providing a cryptographic ”trap-door’ function. The data loss is an intrinsic part of the *RPHash* clustering algorithm that has little adverse effect on its model generation and subsequent recall accuracy.
The only point at which fully qualified vectors are transmitted between compute nodes is during the Phase2 gather stage. This however is not of concern as the data represents an aggregation of individual data. By definition of a centroid it is the average of only the k largest populations of patients. *RPHash* is a clustering algorithm for very large datasets, where the probability of identifying an individual from Phase2 vector information is proportional to the size of the smallest identified cluster.

4 Experimental Approach

The experirimental approach for testing our algorithm will consider 4 major attributes for parallel maching learning algorithms.

4.1 Algorithm Accuracy

A classic analysis of any clustering algorithm is its ability to correctly catagorize unseen data. There are various ways of doing this such as Precision Recall, ROC, and FP/TF with varying applicability to different domain constraints. We plan to provide this data for our algorithm along with various other algorithms performed on large datasets, and in parallel using the MRv2 framework, and distributed systems. A primary resource for comparison will be the industry standard learning algorithms found in the Apache Mahout Machine learning for Hadoop Project. We will use these tools to evaluate both the classification accuracy between RPHash in win/draw/lose contest, as well as the overall speed and scaling complexity of the various algorithms.

4.2 Data

To accurately test our system, a variety of real and synthetic datasets will be incorporated. Our synthetic datasets will consist of varying size and dimension, gaussian clusters, while our real datasets will be acquired from various bioinformatic databases, as well as semantic data from health care resources. An emphasis on health care related data will be used to establish the distributed database security utility of our algorithm.
Our synthetic dataset will consist of k vectors from $\mathbb{R}^d$ choosen from a uniform random distribution. These vectors will form the basis for our cluster centers. For each cluster center an n/k set of vectors will be generated by an additive gaussian noise permutation added to the center vector. Ultimately this will give us n vectors in k Gaussian clusters. From this we will reserve half of the vectors for training, and the other half for testing accuracy.
Our real dataset will consist of genomic data and image SIFT vector data. These sets will give us a respective demonstration of RPHash’s performance on very high dimensional, and relatively high dimensional datasets.

4.3 Security Tests

As our main goal is to show RPHash’s resistance to de-anonymization attacks by use of destructive projection functions, the primary security demonstration will be to show that no single fully qualified record can be associated with the data being sent between Hadoop Nodes.
Auditing security is a difficult task, and is somewhat beyond the scope of this proposal, as such, security demonstrations will be primarily empirical. Known correlated fully qualified vectors and projected vectors will be compared for similarity using the least squares solution of the projection matrix. For orthogonal projection matrices the least squares solution serves as a suitable inverse for the projection function. However, for non-orthogonal projection matrices the least-squares solution is over-determined, and the basises overlap, causing unrecoverable data loss. Our goal in testing is to show that this data loss is enough to make it very difficult to re-associate vectors with any projected data shared among Hadoop Nodes.

5 Related Work

Clustering is a fundemental machine learning algorithm and therefor has been extensively applied to a variety of dataset sizes. Furthermore, due to the inherent scalability issues of many sequentially designed algorithms, parallel algorithms have also been developed to mitigate these issues.

5.1 Density Scan Database Clustering Methods

The first set of parallel clustering algorithms began with density based scanning methods. These methods tend to work well on spacially seperated datasets with relatively low dimension. A very common clustering problem for these types of algorithms would be on geospitial data, such as large maps and image segmentation. The algorithms DBScan [27], Clique [28], and CLARANS [29], respectively, represent a successful progression of the density scanning techniques. Though density scan algorithms are an example of parallel designed algorithms like RPHash, they often show weaknesses in accuracy when scaling the number of dimensions. A proposed solution mentioned below to this problem is PROCLUS [30].

5.2 Random Projection

Another important feature of RPHash is the use of random projection clustering. One of the first algorithms to explore random projection for clustering was Proclus [30]. Proclus used the assumptition that high dimensional data often tends to be overly sparse and feature subset selection would offer a way of compacting the problem. Many subset selection algorithms have been explored [31] [32], they often require computationally expensive gradient methods. This idea, combined with the near accuracy of random projections in preserving discrepancy. From their, the authors suggest an iterative method of medoid searching similar to CLARANS [29]. Though this method is successful in accelerating somewhat larger dimensionality problems ($d = 20$), it is still relatively sequentially bound in its iterative nature.

In addition to PROCLUS, various other methods and analysis have been performed on clustering with random projections that provide some bounds on the convergence and extensibility of random projection clustering. Florescu gives bounds on the scaling and convergence of projected clustering [16]. Their results closely follow the logic of Urruty [15], and find that the number of trial projections, or similarly in our case, projected dimensions, effects the convergence rate exponentially. They also assert the probability of random projection offering a good partitioning plane decreases exponentially as the number of dimensions increases. This suggests RPHash may have to find a way to overcome this issue, or show empirically that the assymptotic behaviour of increasing projected dimensions, overshadows increases in d.

Other clustering by random projection algorithms have been explored that are very similar to our proposed method, but for projections on the line. These so called cluster ensemble approaches [17] [33] [18] use histograms to identify cluster regions of high density much like RPHash. Though, as suggested in Florescu and Urruty, the single dimension approach may plagued by issues of occultation, and exponential convergence as d increases.

6 Proposed Research Schedule

will constitute the majority of our efforts. As it is the algorithm behind the tests in our subsequent aims, it must be completed and tested prior to use. The testing phase against Mahout clustering algorithms however need not be performed at this time. As the development and deployment on commercial cloud computing resources is by in large the longest duration task, we are allotting 6-8 months to complete it. Aim 4. Is an ongoing demonstration of the security tolerance of the system from

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, jan 2008.

[2] X. Ding, L. Zhang, Z. Wan, and M. Gu, "A brief survey on de-anonymization attacks in online social networks," in *Computational Aspects of Social Networks (CASoN), 2010 International Conference on*, pp. 611–615, 2010.

[3] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 111–125, 2008.

[4] G. D. H. E. Gymrek M, McGuire AL and E. Y, "Identifying personal genomes by surname inference," *Science*, no. 339, pp. 321–324, 2013.

[5] "Privacy and progress in whole genome sequencing," report to the president, Presidential Commission for the Study of Bioethical Issues, Washington, October 2012.

[6] f. a. l. carraher, "A parallel algorithm for query adaptive, locality sensitive hash search," 2012.

[7] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *J. ACM*, vol. 27, pp. 831–838, oct 1980.

[8] S. Dasgupta, *The hardness of k-means clustering.* Department of Computer Science and Engineering, University of California, San Diego, 2008.

[9] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *WALCOM: Algorithms and Computation* (S. Das and R. Uehara, eds.), vol. 5431, ch. Lecture Notes in Computer Science, pp. 274–285, Springer Berlin Heidelberg, 2009.

[10] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 274–281, 2001.

[11] S. S. Vempala, *The Random Projection Method.* DIMACS Series, American Mathematical Society, 2004.

[12] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *in Knowledge Discovery and Data Mining*, pp. 245–250, ACM Press, 2001.

[13] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets.* Cambridge University Press, 2011.

[14] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?," in *In Int. Conf. on Database Theory*, pp. 217–235, 1999.

[15] T. Urruty, C. Djeraba, and D. Simovici, "Lecture notes in computer science," in *Advances in Data Mining. Theoretical Aspects and Applications* (P. Perner, ed.), vol. 4597, ch. Clustering by Random Projections, pp. 107–119, Springer Berlin Heidelberg, 2007.

[16] I. Florescu, A. Molyboha, and A. Myasnikov, "Scaling and convergence of projection sampling," 2009.

[17] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach,"

[18] R. Avogadri and G. Valentini, "Fuzzy ensemble clustering based on random projections for dna microarray data analysis," *Artificial Intelligence in Medicine*, vol. 45, no. 2, pp. 173–183, 2009.

[19] A. Vattani, "k-means requires exponentially many iterations even in the plane," in *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, (New York, NY, USA), pp. 324–332, ACM, 2009.

[20] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, (New York, NY, USA), pp. 1186–1195, ACM, 2006.

[21] H. Cohn and A. Kumar, "Optimality and uniqueness of the leech lattice among lattices," tech. rep., 2004.

[22] S. Dasgupta, "Experiments with random projection," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, UAI'00, (San Francisco, CA, USA), pp. 143–151, Morgan Kaufmann Publishers Inc., 2000.

[23] A. Vardy, "Even more efficient bounded-distance decoding of the hexacode, the golay code, and the leech lattice," *Information Theory, IEEE Transactions on*, vol. 41, no. 5, pp. 1495–1499, 1995.

[24] A. Andoni, *Nearest Neighbor Search: the Old, the New, and the Impossible.* PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, September 2009.

[25] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2010.

[26] M. Chowdhury, M. Zaharia, and I. Stoica, "Performance and scalability of broadcast in spark,"

[27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *KDD*, vol. 96, pp. 226–231, 1996.

[28] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*, vol. 27. ACM, 1998.

[29] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 5, pp. 1003–1016, 2002.

[30] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMOD Record*, vol. 28, pp. 61–72, 1999.

[31] R. Kohavi and D. Sommerfield, "Feature subset selection using the wrapper method: Overfitting and dynamic search space topology.," in *KDD*, pp. 192–197, 1995.

[32] P. S. Bradley, O. L. Mangasarian, and W. N. Street, "Clustering via concave minimization," *Advances in neural information processing systems*, pp. 368–374, 1997.

[33] M. Al-Razgan and C. Domeniconi, "Weighted clustering ensembles," in *Proceedings of the SIAM international conference on data mining*, pp. 258–269, 2006.