Random Hyperplane Projection using Derived Dimensions*

Konstantinos Georgoulas
Athens University of Economics and Business
76 Patission Street
Athens, Greece
kgeorgou@aueb.gr

Yannis Kotidis
Athens University of Economics and Business
76 Patission Street
Athens, Greece
kotidis@aueb.gr

ABSTRACT

Computing the similarity between data objects is a fundamental operation for many distributive applications such as those on the Word Wide Wed, in Peer-to-Peer networks or even in Sensor Networks. Locality Sensitive Hashing (LSH) has been recently proposed in order to reduce the number of bits that need to be transmitted between sites in order to permit evaluation of different similarity functions between the data objects. In our work we investigate a particular form of LSH, termed Random Hyperplane Projection (RHP). RHP is a data agnostic model that works for arbitrary data sets. However, data in most applications is not uniform. In our work, we first describe the shortcomings of the RHP scheme, in particular, its inefficiency to exploit evident skew in the underlying data distribution and then propose a novel framework that automatically detects correlations and computes an RHP embedding in the Hamming cube tailored to the provided data set. We further discuss extensions of our framework in order to cope with changes in the data distribution or outliers. In such cases our technique automatically reverts to the basic RHP model for data items that can not be described accurately through the computed embedding. Our experimental evaluation using several real datasets demonstrates that our proposed scheme outperforms the existing RHP algorithm providing up to three times more accurate similarity computations using the same number of bits.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Locality Sensitive Hashing, Similarity, Sensor Networks

*This work was partially supported by the Basic Research Funding Program, Athens University of Economics and Business.

1. INTRODUCTION

Computing the similarity between data items is a fundamental operation for many applications. Quite often, the data in question is not available at a central location, but is rather dispersed between sites. For example, pervasive applications are increasingly supported by networked sensory devices that interact with people and themselves in order to provide the desired services and functionality. Because of the unattended nature of many installations and the inexpensive hardware used in the construction of the sensors, nodes often generate imprecise individual readings due to interference or failures [11]. Recent proposals consider mechanisms for cleaning sensor readings by identifying and, possibly, removing outliers [2, 5, 6]. At the core of these techniques lies the need of a network-friendly mechanism for computing the similarity between recent measurements of distant nodes. Sensory data, collected by nodes needs to be processed and understood in a decentralized manner, so as to avoid depleting the limited resources available at the nodes. A central collection of sensory data is not feasible nor desired, since it results in high energy drain, due to the large number of transmitted messages. Moreover, data in most applications, is continuously collected by the sensor nodes, increasing the need to develop techniques that limit the amount of data transmitted [3, 4].

The need to perform similarity tests between data is also evident in distributed applications such as those on the Word Wide Wed or in Peer-to-Peer networks. Data sharing, management and query processing in such settings requires techniques that reduce the need to transfer or duplicate data (or meta-data) amongst sites. As an example, recent proposals consider peer-to-peer architectures for enabling advanced query processing, when data is horizontally distributed amongst sites [14]. In order to facilitate efficient query routing and processing via an overlay network topology, data clustering techniques are implemented. These techniques often require the evaluation of similarity metrics between data stored at different peers. Likewise, when integrating data stores over the Web, understanding not only schema but also data (dis)-similarity is fundamental for the success of any integration task.

All of the aforementioned applications, while diverse in their assumptions and architecture share the need of a primitive operation that will allow the assessment of similarity between descriptions of data that are located at different sites. These descriptions need to be (i) easily computed from the data attributes, so as to reduce processing cost, and (ii) compact in size, so as to permit their network transmission, instead of the original data, for similarity testing.

In our work, we provide a framework that addresses both these challenges. Our techniques assume a generic description of data as multidimensional points and allow the computation of common similarity metrics such as the cosine coefficient and the correlation coefficient. We adopt the Random Hyperplane Projection (RHP) framework [1, 9], a novel dimensionality reduction technique based on locality sensitive hashing (LSH) [1, 10] that is used to transform each d-dimensional point into a much shorter bitmap of n bits. This encoding is performed independently at each site for its local data. RPH is a powerful technique that trades accuracy for bandwidth, by simply varying the desired level of dimensionality reduction. The loss of accuracy comes from the projection of the original data into a space of lower dimensionality, however, it can be easily controlled by varying the desired length of the bitmaps and through a boosting process that utilizes multiple (shorter) bitmaps [6] for computing the similarity of the data objects.

The main drawback of the RHP mapping is that, it assumes a uniform distribution of the data in the d-dimensional space. Nevertheless, data in real applications is unlikely to be uniform. As an example, when sensors monitor physical quantities like humidity, pressure or light, the collected data values are typically skewed, reflecting the conditions in the monitored area. In Web or P2P applications, data on peers is usually clustered into a few thematic areas that reflect the user's interests. As will be demonstrated, the uniform assumption of a typical data-agnostic RHP encoding scheme, severely limits its performance. In this work we propose a dimensionality reduction framework that takes into account skew that is often evident in the data distribution. Our techniques deliberately alter the way data is mapped into a lower-dimensionality space so as to increase the accuracy of the produced bitmaps. Moreover, our algorithms retain the advantages of the basic RHP scheme, in particular its simplicity in producing the mappings and subsequently computing the similarity of the original data objects based on them. This is necessary in applications where nodes have limited processing capabilities (as in sensor nodes) or when large volumes of data need to be processed (as in Web or P2P applications).

The main advantage of our proposed framework is that it detects and exploits skew or correlations in the underlying data distribution. Similar to RHP, a n-dimensional representation of the data is computed in the Hamming cube. However, our framework also computes an extended mapping into m-additional dimensions. This mapping is derived based on simple precomputed statistics obtained from a sample of the data and the available embedding on the n-dimensional Hamming cube. In this way, our method manages to project the data items into a higher dimensionality space (n+m dimensions), while still using n bits for the data encodings. The additional, derived, m dimensions are utilized when computing the similarity between the data items, resulting in this way in more accurate evaluations. The new framework is termed $RHP(n, m)$ in order to distinguish it from the basic RHP technique.

The contributions of this paper can be summarized as follows:

- We introduce $RHP(n, m)$, an LSH data reduction framework that supports various popular similarity measures used in different application areas. Examples of such measures include, but are not limited to, the cosine coefficient, the correlation coefficient, or the Jaccard coefficient. Unlike the original RHP technique that is oblivious to the underlying data, $RHP(n, m)$ utilizes prior knowledge of the data distribution in order to produce, with the same space, more accurate descriptions and, thus, allows a more accurate computation of the similarity based on them.

- We describe a novel process for capturing the distribution of the data and accordingly alter the RHP mappings. This is achieved by computing a few intuitive statistics using a sample obtained not from the data items (as this would negate the benefits of the whole framework) but rather from much shorter RHP encodings.

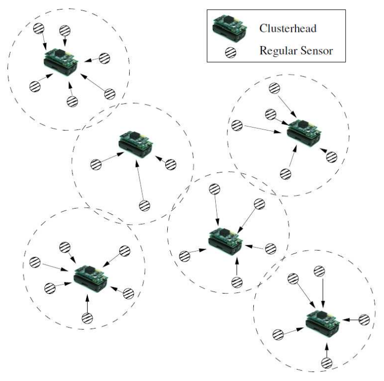

Figure 1: Computing Outliers ([6]): motes transmit RHP bitmaps describing their latest d measurements to their clusterheads. Each clusterhead computes a local list of potential outliers based on pair-wise similarity tests of all motes in its cluster. Local lists are comminucated between clusterheads in order to compute a final global list of outliers (not shown in Figure).

- We introduce techniques that detect data items that can not be accurately described using the available statistics. For such items, our algorithms automatically fall back into using the basic $RHP(n)$ scheme that is oblivious to the data characteristics.

- We present a detailed experimental analysis of our techniques for a variety of real data sets. Our results demonstrate that our methods can reliably compute the similarity between data objects and consistently outperform the standard RHP scheme.

This paper proceeds as follows. In Section 2 we present an application of our framework for computing outliers in wireless sensor networks. In Section 3 we discuss related work. Section 4 presents the basic Random Hyperplane Projection (RHP) technique and discusses its advantages and shortcomings. In Section 5 we formally present our framework. Section 6 presents our experimental evaluation, while Section 7 provides concluding remarks.

2. MOTIVATIONAL EXAMPLE

In our recent work we introduced a distributed framework for computing outliers in wireless sensor networks based on RHP [6]. Our method assumes a clustered network organization depicted in Figure 1. Regular sensor motes compute RHP encodings from their latest d measurements. These encodings are transmitted to their clusterheads, which can estimate the similarity amongst the latest values of any pair of motes within its cluster by comparing their bitmaps. Based on the performed similarity tests and a desired minimum support specified by the application, each clusterhead is able to compute a potential list of local outliers. These lists are then communicated among the clusterheads in order to compute the final list of outliers that is reported to the application. At the core of this process lies the requirement to accurately evaluate the similarity of d-dimensional data vectors (containing the measurements obtained by the sensors) in a network-friendly manner using the much shorter RHP bitmaps instead. The techniques we present in this paper can be directly applied in this application. All is required is to replace the original RHP bitmaps with the proposed $RHP(n, m)$ encodings. This would result in increased accuracy

Symbol	Description
x,y	data items described as d-dimensional vectors (points)
$\theta(x, y)$	the angle between vectors x,y
$lsh(x)$	the bitmap encoding produced after applying RHP to x
n	RHP bitmap length
r_i	ith random d-dimensional vector
$h_{r_i}(x)$	hash function for r_i applied on data item x
$D_h(lsh(x), lsh(y))$	the hamming distance between RHP bitmaps
$P(j\|i)$	probability that $h_{r_j}(x)$=1 when $h_{r_i}(x)$=1
$P(j\|\neg i)$	probability that $h_{r_j}(x)$=1 when $h_{r_i}(x)$=0
$expLsh(x)$	(n+m)-dim vector. First n values are obtained from $lsh(x)$. Remaining m values are derived using conditional probabilities $P_{j\|i}$, $P_{j\|\neg i}$

Table 1: Notation used in this paper

when computing the similarity between sensory data, resulting in fewer false positive/negative cases of outlier identification.

3. RELATED WORK

The Locality Sensitive Hashing (LSH) scheme that we extend in this paper was initially introduced in [9] to provide solutions to the MAX-CUT problem. Since then, LSH has been applied in many applications including similarity estimation [1] and clustering [13]. Moreover, it was utilized in data structures intended to support approximate nearest neighbor queries [10] or indexing techniques for set value attributes [8].

In our work, we investigate techniques that increase the accuracy of LSH applied in similarity estimation. The Random Hyperplane Projection scheme [9] that we consider can be used to compute popular similarity metrics such as the cosine coefficient, the correlation coefficient and the Jaccard index. The correlation coefficient and the Jaccard index have been recently considered and evaluated for detecting outliers in sensor networks [5, 6, 15]. The cosine similarity has been used in diverse applications such as IP traffic monitoring [7] and computing the similarity between documents [12]. Our techniques can be used in any of the aforementioned applications for extending the accuracy of the similarity evaluations between data items through their LSH encodings, while limiting the amount of data that needs to be transferred between remote sites.

4. PRELIMINARIES

4.1 The RHP framework

We now present the basic locality sensitive hashing scheme that our framework extends. The notation used in our discussion is summarized in Table 1. The corresponding definitions are presented in appropriate areas of the text.

A *Locality Sensitive Hashing scheme* is defined in [1] as a distribution on a family F of hash functions that operate on a set of data items, such that for two data items x, y:

$$P_{h \epsilon F}[h(x) = h(y)] = sim(x, y) \tag{1}$$

where $sim(x, y) \epsilon [0, 1]$ is some similarity measure.

In our framework we utilize a particular form of LSH termed Random Hyperplane Projection (RHP) [1, 9]. We assume a collection of data described in the d-dimensional space. In RHP, we generate a family of hash functions as follows. We produce a spherically symmetric random vector r of unit length from this d dimensional space. Using r, we define a hash function h_r as:

$$h_r(x) = \begin{cases} 1 & \text{,if } r \cdot x \geq 0 \\ 0 & \text{,if } r \cdot x < 0 \end{cases} \tag{2}$$

i.e. $h_r()$ evaluates to 1 for all data items whose dot product with r is positive and to 0 for the rest of the data. It is easy to see ([16]) that for any two vectors x, y

$$P[h_r(x) = h_r(y)] = 1 - \frac{\theta(x, y)}{\pi} \tag{3}$$

If we repeat this process using n random vectors $r_1, \ldots, r_n$, an input data item x is mapped into a bitmap $lsh(x)$ of length n. Bit i in this bitmap is the evaluation of $h_{r_i}(x)$.

Let x and y be two input data items and $lsh(x)$, $lsh(y)$ their RHP bitmaps respectively. Based on Equation 3 it follows that

$$\frac{\theta(x, y)}{\pi} = \frac{D_h(lsh(x), lsh(y))}{n} \tag{4}$$

$D_h(lsh(x), lsh(y))$ in the above formula denotes the hamming distance of the produced bitmaps. This Equation states that the number of bits that differ in the RHP encodings of vectors x and y is proportional to their angle. Solving the formula for $\theta(x, y)$ allows us to estimate the angle between the two vectors from their RHP encodings.

From the angle computation, one can trivially derive the cosine similarity $cos(\theta(x, y))$ between x and y. Moreover, let $E(x)$ denote the mean value of vector x. The correlation coefficient $corr(x, y)$ between x and y can then be computed as $corr(x, y)$ = $corr(x - E(X), y - E(y))$=$cos(\theta(x - E(x), y - E(y))$ [6]. Thus, using the RHP bitmaps we can also compute the correlation coefficient of x and y. Both these metrics are fundamental in assessing the similarity between data items. For instance, the cosine similarity is used in [7] in evaluating the similarity between network traffic patterns in IP networks. Similarly, the correlation coefficient has been recently used in detecting outliers in measurements obtained from sensor networks [5]. The LSH scheme is extended in [8] to further support the popular Jaccard index.

4.2 Benefits and Shortcomings of RHP

The basic RHP scheme is an intuitive method for reducing the size (dimensionality) of the input data items, while retaining the ability to compute the angle (similarity) between them. Moreover, the RHP scheme works easily in distributed settings. What is required, is that all sites (sensor nodes, peers, etc) utilize a common seed value in order to generate locally the same family of random vectors r_i. (Thus, there is no need to transfer the random vectors between sites at a pre-processing step.) Then, the $lsh()$ encodings can be constructed independently and communicated as needed in order to compute similarity between data objects stored in remote sites. The benefit of applying RHP is that much fewer bits need to be transferred in such cases. For example, assuming a typical 32 bit internal representation of real values, the reduction ratio RR obtained by using RHP bitmaps of length n instead of the actual data objects is:

$$RR = \frac{\text{size of original data description}}{\text{size of RHP bitmap}} = \frac{32 * d}{n} \tag{5}$$

Thus, the benefits of RHP increase linearly with the volume of data that needs to be transmitted. Another characteristic that increases the suitability of RHP for restricted environments is that its encodings are computed in a straightforward manner. All that is required is to compute the sign of simple linear equations (dot products). In case of severe memory constraints, a site does not need to store the random vectors r_i locally. Using the common seed, the random vectors can be generated on the fly for the computation of each dot product. Thus, the technique requires $O(d)$ space and $O(n * d)$ time per item.[1] Both requirements are rather modest. Computing

[1]If all random vectors are materialized, the space requirements increase to $O(n * d)$.

the angle between two data objects from their encoding through Equation 4 entails the computation of the hamming distance between bitmaps $lsh(x)$, $lsh(y)$, a process that is done efficiently in most platforms by XORing the bitmaps and counting the ones in the result.

The shortcomings of the basic RHP scheme stem from the fact that it requires a family F of random vectors r_i that are uniformly distributed in the d-dimensional space. When the distribution of the data objects is not uniform, this results in under-utilizing many members of F. Figure 2 provides an intuition of the works of RHP in two-dimensions. For the sake of this example, let us assume that all data objects (two dimensional points) fall in the area (slice) denoted as D in the Figure. It is easy to see that for all random vectors r_i that do not belong in one of the two "orthogonal" slices O_1 and O_2 in the Figure, their dot product with x always has the same sign. All such random vectors do not contribute in computing the angle between two objects x and y (from slice D), since the corresponding bits will either be both set (one) or clear (zero). Only random vectors from slices O_1 and O_2 may produce different results for x and y. When data skew increases, slices D, O_1 and O_2 become thinner and, thus, the percentage of "useful" random vectors r_i decreases proportionally. Similar arguments apply in higher dimensions. This simplified example demonstrates that for data that is not uniformly distributed in R^d, often many of the bits used in the $lsh(x)$ encoding cannot contribute towards computing the angle of the vectors. This means that out of the n bits that we transmit, typically only a few of those are helpful in computing the similarity between the data. Unfortunately, without knowing before-hand the values of x and y it is not possible to decide, which of the random vectors are useful and which are not.

One may be tempted to devise families of random vectors that are tailored to a particular data set. In the example of Figure 2, this can be easily achieved by placing all random vectors r_i within slices O_1 and O_2. While this is possible in two-dimensions (Formula 4 needs to be modified accordingly in such a case), there is no obvious way to apply this process in higher dimensions. Notice that in such cases, each pair of d-dimensional vectors x, y defines a different plane, and therefore, hyper-slices O_1, O_2 are defined differently for each input pair x, y in consideration.

5. OUR $RHP(N, M)$ SCHEME

5.1 Overview

We now discuss our new LSH scheme that alleviates the shortcomings of RHP while it retains its benefits. To distinguish it from our proposed framework, we will refer to the basic RHP process as $RHP(n)$. As already noted, quite often many of the n random vectors employed in $RHP(n)$ do not contribute in the computation of the similarity, as they result in similar bits (one or zero) for many data items. A key idea of our framework is to detect and exploit such correlations between the random vectors r_i by considering an extended family of n+m random vectors (for $m \geq 0$). As in $RHP(n)$, this family is computed using a common seed value. We will deliberately distinguish two types of random vectors: materialized and derived. Materialized random vectors contribute to the encoding $lsh(x)$ by producing a bit value based on Equation 2. Derived random vectors r_j are not used in constructing the bitmap. As will be explained, the value of their hash function $h_{r_j}(x)$ is estimated using the $h_{r_i}(x)$ values of the materialized r_is and some precomputed statistics. In total, there are n materialized random vectors and m derived ones. Thus, the $lsh(x)$ encoding of x in our framework will still contain exactly n bits, entailing the same

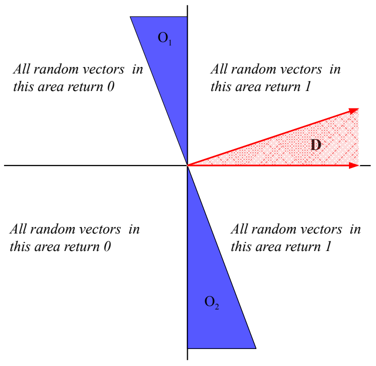
O1
All random vectors in this area return 0
All random vectors in this area return 1
D
All random vectors in this area return 0
All random vectors in this area return 1
O2

Figure 2: Performance of $RHP(n)$ when all data falls into area D. Only random vectors in the shaded areas O_1 and O_2 can help distinguish between different data items.

construction and communication overhead as in $RHP(n)$. We will refer to our proposed framework as $RHP(n, m)$.

The difference in the new scheme is in the way we decode the bitmaps for computing the angle between two data items x and y. During the decoding process all n+m random vectors are utilized, resulting in increased accuracy compared to $RHP(n)$ that only utilizes n random vectors. In our framework, each derived random vector r_j is associated with exactly one materialized random vector r_i that we refer to as its "representative". The representative random vector r_i is used in order to compute the probability that the j-th bit that corresponds to $h_{r_j}(x)$ (which is not available in the $lsh()$ encoding) would be set. Since our main focus is to retain the benefits of the RHP scheme and in particular (i) its simplicity in computing the angle between the data objects and (ii) the small-space requirements of the decoding process, we will utilize two simple statistics for each derived random vector in the decoding step. Let $P_{j,i}$=$P[h_{r_j}(x) = 1 | h_{r_i}(x) = 1]$ denote the probability that the hash function $h_{r_j}(x)$ evaluates to one when $h_{r_i}(x)$=1 over all possible data items x in our data set. Similarly, let $P_{j,\neg i}$=$P[h_{r_j}(x) = 1 | h_{r_i}(x) = 0]$ denote the probability that the hash function $h_{r_j}(x)$ evaluates to one when the hash value for the i-th random vector is zero. During the decoding process we utilize these pre-computed probabilities in order to estimate the value of the j-th bit.

More formally, our decoding process computes an intermediate extended representation of n+m values described as a vector $expLsh(x)$. The coordinates of this vector are computed as follows:

- If the j-th random vector is materialized, the value used in the j-th coordinate of $expLsh(x)$ is the j-th bit that has been computed in $lsh(x)$.
- If the j-th random vector is derived, then the value of the j-th coordinate in $expLsh(x)$ is computed from the representative r_i of r_j as $P_{j|i}$, if the i-th bit in $lsh(x)$ is set or as $P_{j,\neg i}$ otherwise.

Intuitively, the $expLsh(x)$ encoding is an approximation of the $lsh(x)$ bitmap that would be obtained if all random vectors were materialized (as in $RHP(n + m)$). Since, the values of $h_{r_j}(x)$ are not available for derived random vectors, we use the conditional probabilities $P_{j,i}$ and $P_{j,\neg i}$ and the available bits from their representatives.

The angle between two input data items x and y is computed by manipulating their $expLsh()$ representations. Let $expLsh_i(x)$

denote the i-th coordinate of vector $expLsh(x)$. Based on its construction, $expLsh_i(x)$ denotes the probability that $h_{r_i}(x)$=1. Thus, the probability that the i-th hash function $h_{r_i}()$ produces different results for inputs x and y based on their available $lsh(x)$ and $lsh(y)$ encodings is

$$\begin{aligned} diff_i(lsh(x), lsh(y)) = \\ expLsh_i(x) * (1 - expLsh_i(y) + \\ +(1 - expLsh_i(x)) * expLsh_i(y) \end{aligned} \quad (6)$$

where $diff_i(lsh(x), lsh(y)) \in [0, 1]$. Then, the expected hamming distance of the n+m hash values for x and y can be estimated as

$$expD_h(lsh(x), lsh(y)) = \sum_{i \in [0, n+m)} diff_i(lsh(x), lsh(y)) \quad (7)$$

Please observe that $expD_h(lsh(x), lsh(y)) = D_h(lsh(x), lsh(y))$ for m=0. Based on Equation 4 the angle between vectors x and y is computed as

$$\theta(x, y) = \frac{expD_h(lsh(x), lsh(y))}{n + m}\pi \quad (8)$$

The described decoding process requires $O(n+m)$ space and time. In terms of the communication cost, $RHP(n, m)$ produces bitmaps of length n, as in $RHP(n)$.

5.2 Sampling the Data Distribution

In order to compute the required statistics that our method needs, we employ a sampling process in order to obtain a random sample S of the data set. A key point in our work is that we do not sample from the original dataspace but rather from the RHP encodings of the data. In particular, using the common seed value we generate the $RHP(n + m)$ encodings of S. In case of data stored in remote sites, this means that only the $lsh(x)$ bitmaps of length n+m each are transmitted towards a central location. Let $|S|$ denote the cardinality of the sampled data, then this process requires transmitting $|S|(n + m)$ bits. The sampled RHP encodings are stored in a two dimensional $|S| \times (n + m)$ array $sLSH$. The rows of the array correspond to different $lsh(x)$ representations, while its columns to the random vectors used. Thus, $sLSH[i, j]$ denotes the j-th bit of the lsh bitmap for the i-th data item.

An important aspect of the described sampling process is that it does not require the transmission of the original data items but rather their $RHP(m + n)$ encodings. As a consequence, in an application like the one discussed in Section 2, we use the basic $RHP(n + m)$ framework for the first few similarity tests, exploiting the gathered bitmaps in order to construct the sample. This results in an overhead of m bits per item compared to $RHP(n)$, while the sample is constructed.

5.3 Choosing Amongst Random Vectors

Recall (Equation 4) that a random vector r_i contributes to computing the angle between data items x and y, when $h_{r_i}(x) \neq h_{r_i}(y)$. Based on this observation we compute the *utility* of random vector r_i as

$$utility_i = \sum_{0 \le x < |S|-1, x+1 \le y < |S|} |sLSH[x, i] - sLSH[y, i]| \quad (9)$$

Thus, the utility of r_i measures the number of occasions random vector r_i contributes bits that differ over all possible pairs x, y in the sample.

In addition to choosing random vectors with high utility scores, we also want to materialize random vectors that can be used to predict the behavior of non-materialized random vectors (Equation 8).

Algorithm 1 GREEDY ALGORITHM

```
Require: $(n, m, \{r_i | i = 1..(n + m)]\}, T)$
 1: {Initially all $r_i$s are candidates for materialization}
 2: $Cand$={$r_i | i = [1..(n + m)]$}
 3: $Mat$=$\emptyset$ {Materialized random vectors}
 4: $Der$=$\emptyset$ {Derived random vectors}
 5: while ($|Mat|$<$n$) AND ($Cand \neq \emptyset$) do
 6:     {Select $r_i$ with highest utility score}
 7:     $k$=$argmax_{i \in Cand}(utility_i)$
 8:     $Mat = Mat \cup \{r_k\}$ {Mark $r_k$ as materialized}
 9:     $Cand$=$Cand$-{$r_k$} {Remove from candidate list}
10:     {Remove strongly correlated (to $r_k$) random vectors}
11:     for $r_i \in Cand$ do
12:         if $|corr_{i,k}| \geq T$ then
13:             $Cand$=$Cand$-{$r_i$}
14:         end if
15:     end for
16: end while
17: {Remaining $r_i s$ are marked as derived}
18: $Der$={$r_i | i = [1..(n + m)]$} - $Mat$
19: for $r_j \in Der$ do
20:     $i$=$argmax_{k \in Mat}(|corr_{k,j}|)$
21:     $Representative(j) = i$ {Mark as representative}
22: end for
```

Given two random vectors r_i and r_j the columns i and j of array $sLSH$ depict the behavior of these random vectors (i.e., the values of their respective hash functions) for the sampled data. Let $sLSH[., i]$ denote the i-th column of $sLSH$. Each such column is a bitmap of length $|S|$. A standard way to assess the correlation between the values of these bitmaps is to compute their correlation coefficient $corr_{i,j}$:

$$corr_{i,j} = \frac{cov(sLSH[., i], sLSH[., j])}{\sigma_{sLSH[.,i]}\sigma_{sLSH[.,i]}} \quad (10)$$

where $cov()$, σ are the covariance and standard deviation functions respectively. A strong (positive or negative) correlation between columns $sLSH[., i]$ and $sLSH[., j]$ indicates that random vector r_i is a good candidate for representing r_j and vise-versa. Thus, we want $corr_{i,j}$ be near +1 or -1. On the contrary when $|corr_{i,j}|$ is close to zero, then there is no evident connection (in the sample) in the behavior of the two random vectors.

Based on these observations we propose a simple greedy algorithm for selecting n random vectors to be materialized (out of the n+m available choices). The algorithm is presented in Algorithm 1 and proceeds as follows. Initially all n+m random vectors are candidates for materialization (set $Cand$, Line 2). At each step, the algorithm selects from set $Cand$ the random vector r_k with the highest utility score and places it in the materialized set Mat (Lines 6-8). When random vector r_k is selected for materialization, we remove from set $Cand$ all random vectors r_i that have a strong correlation (based on parameter T) with r_k (Lines 10-16). The intuition is that the hash values for these r_js can be easily estimated using conditional probabilities $P_{j|i}$ and $P_{j|\neg i}$. A typical value of T is 0.95 in our implementation. The algorithm repeats this step, until n random vectors have been selected. At a final phase (Lines 17-22) the algorithm selects the representative of each derived random vector (the remaining m random vectors not in set Mat at the end of the selection process) based on the absolute values of the correlation coefficients $corr_{k,j}$ between a materialized random vector r_k and a derived random vector r_j. The running time of the

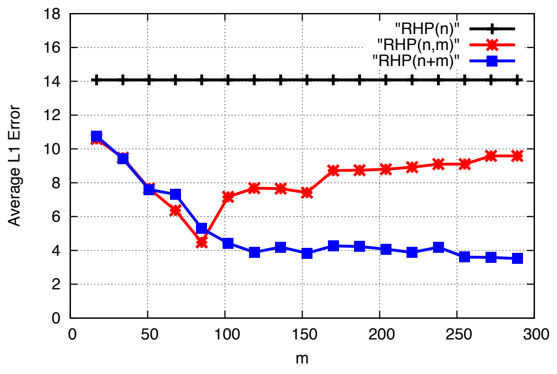

Figure 3: Sensitivity to m, NBA data set (n=34). $RHP(n)$, $RHP(n,m)$ **use 34 bits/data item.** $RHP(n+m)$ **uses 34+m bits per data item.**

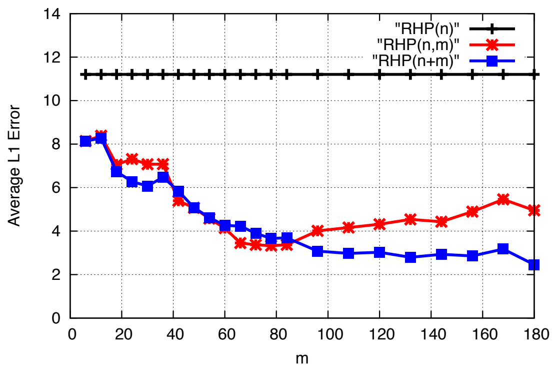

Figure 4: Sensitivity to m, HOUSE data set (n=12). $RHP(n)$, $RHP(n,m)$ **use 12 bits/data item.** $RHP(n+m)$ **uses 12+m bits per data item.**

algorithm is $O(n \times (n+m))$. We note that in the applications of interest, n, m take small values, typically, less than 100.

After the selection of the representative random vectors, it is straightforward to compute probabilities $P_{j|i}$ and $P_{j|\neg i}$ from the corresponding columns $sLSH[., i]$ and $sLSH[., j]$. Then, array $sLSH$ can be discarded. In the end of this process we have:

- A set of n random vectors Mat denoted as materialized.
- A set of m random vectors Der denoted as derived.
- The representative of each derived random vector (m values in total).
- The conditional probabilities ($2 \times m$ values in total) required for estimating the angles between data items using Equation 8.

Using this information (of size $O(n+m)$) any site is able to compute the similarity of two data items from their $RHP(n,m)$ bitmaps.

5.4 Handling Evolving Data Sets

The proposed $RHP(n,m)$ framework makes use of precomputed statistics in order to boost the accuracy of the standard random hyperplane projection scheme. A natural question that arises, is whether our techniques will be able to adapt to (transient or permanent) changes in the characteristics of the data sources. In this section we introduce techniques that detect data items that cannot be accurately described using the available statistics. In such a case, our algorithms fall back into using the basic $RHP(n)$ scheme that is oblivious to the data characteristics.

In order to distinguish the encodings of the discussed RHP schemes, we will use the notation $lsh_k(x)$ to denote the bitmap obtained by $RHP(k)$. In our original $RHP(n,m)$ framework we compute and communicate $lsh_{n,m}(x)$ bitmaps (of length n). Given the common seed value, a local site can further compute the $RHP(n+m)$ encoding $lsh_{n+m}(x)$. Let $\hat{x}_{n,m}$ denote the approximation of data item x provided through its $lsh_{n,m}(x)$ embedding and, similarly $\hat{x}_{n+m}$ denote the approximation obtained via $lsh_{n+m}(x)$. Using Equation 8, we can easily compute the angle $\theta(\hat{x}_{n,m}, \hat{x}_{n+m})$ from the corresponding bitmaps. Recall that $RHP(n+m)$ (the basic scheme using n+m bits) is oblivious to the data distribution and more accurate than $RHP(n)$ since it utilizes additional dimensions towards creating its embedding in the Hamming cube. Based on this observation, for each data item x we simply test whether our $RHP(n,m)$ approximation $\hat{x}_{n,m}$ of x is sufficiently close to the approximation $\hat{x}_{n+m}$ provided by $RHP(n+m)$, i.e.

$$\theta(\hat{x}_{n,m}, \hat{x}_{n+m}) \leq \theta_\phi \tag{11}$$

where θ_ϕ is a threshold denoting the tolerance of our application. For each data item x that fails this test, we will transmit instead the $lsh_n(x)$ obtained using $RHP(n)$, augmented with an additional bit that informs the site that will decode the embedding that the standard $RHP(n)$ process has been used. Given two encodings $lsh_{n,m}(x)$, $lsh_{n,m}(y)$ (resp. $lsh_n(x)$, $lsh_n(y)$), we compute the angle of the original data items using Equation 8 (resp. Equation 4). In order to compare $lsh_n(x)$, $lsh_{n,m}(y)$, we first obtain $expLSH(y)$ as already described, and then we compare the bitmaps via Equation 8 using only the hash values of the random vectors utilized in $RHP(n)$ and setting m=0.

The advantages of this simple extension are twofold. First, it can easily detect transient readings that differ significantly from the sampled data used in learning the conditional probabilities $P_{j|i}$ and $P_{j|\neg i}$ and fall back into using the basic $RHP(n)$ scheme for them. Second, when a large number of input data items fail the test of Equation 11, this can be used to trigger a new sampling process in order to compute a new set of materialized random vectors using Algorithm 1 for the updated data distribution.

6. EXPERIMENTS

In our experimental evaluation we utilize the following four real datasets.

- **TEMPERATURE:** This data set contains temperature information obtained by sensor nodes at the Intel Labs [5]. We used non-overlapping windows of 32 epochs to generate 912 32-dimensional records of sensory measurements.
- **HUMIDITY:** This data set contains humidity measurements obtained by sensor nodes at the Intel Labs [5]. We used non-overlapping windows of 32 epochs to generate 720 32-dimensional records of sensory measurements.
- **NBA:** This data set contains 21,384 17-dimensional records each representing a player's performance per year. Attributes include, minutes played, points, offensive/defensive rebounds, assists, etc.
- **HOUSE:** This data set consists of 100,000 6-dimensional records containing real-estate information from the United

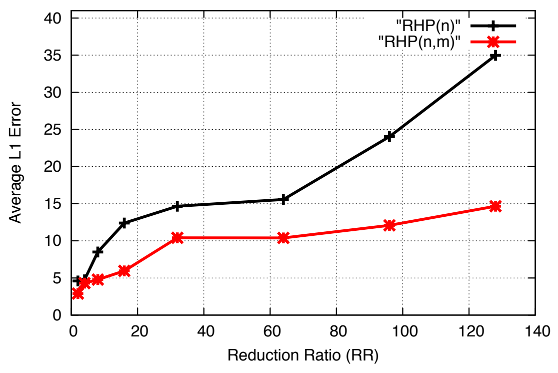

Figure 5: NBA Dataset

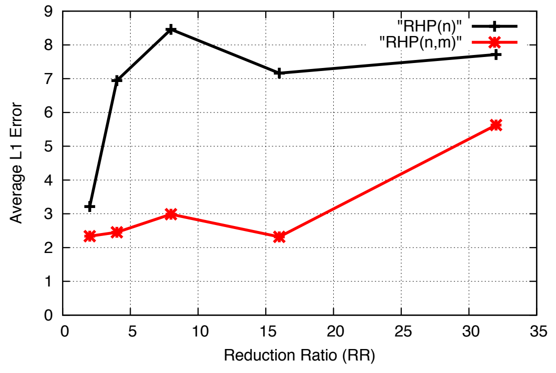

Figure 6: HOUSE Dataset

States (data available at zillow.com). Features include price, number of rooms, area etc.

In order to assess the performance of our proposed framework we perform several experiments and present the average L_1 error in computing the angle between different pairs of records x, y. The L_1 error is defined as

$$error_{L_1}(x, y) = |\theta(x, y) - \theta(lsh(x), lsh(y))| \tag{12}$$

where $\theta(x, y)$ denotes the real angle between the data items and $\theta(lsh(x), lsh(y))$, the estimate we obtain from the LSH encodings of the respective method. Please notice that from the L_1 error it is trivial to obtain the estimation error in computing the cosine similarity. We also ran experiments using the correlation coefficient as the similarity metric however they are omitted due to lack of space. In all runs, we use a 10% sample as an input to Algorithm 1 (with the exception of the much larger HOUSE dataset for which a 2% sample was used) and for computing the conditional probabilities. We then evaluated the accuracy of the discussed techniques using a different sample of 100 points and evaluating the angle estimates over all 4950 possible pairs between them.

In the experiments of Figures 3, 4 we evaluate the performance of our technique when we vary m, the number of derived random vectors. Parameter n was set to 34 and 12 for the NBA and the HOUSE dataset respectively, so that in both experiments we achieve a reduction ratio of 16:1 (Equation 5). We can see that for all depicted values of m, the $RHP(n, m)$ framework outperforms $RHP(n)$ (which uses the same space per item) up to a factor of 3. More importantly, the following trend seems to appear. Initially, increasing the value of m from 1,2,... results in better estimates. However, after we exceed a certain threshold, a further increase in m results in worse performance. This happens because the additional derived random vectors we introduce are not strongly correlated to the materialized set of random vectors, and thus, their behavior cannot be accurately described by the conditional probabilities $P_{j|i}$ and $P_{j|\neg i}$. We observed the same trend for different values of n and for the other data sets. The shape of the curves in these Figures suggests that, for an application-selected reduction ratio (equivalently a given value of n) we can obtain a good selection m^* by performing a quick search over a small range of values for parameter m, using the sample. In all experiments reported in the rest of this Section, we used this process in the range $m : (0, 8n)$ in order to select the best m in each run.

In Figures 3, 4 we also depict the performance of $RHP(n + m)$ that transmits m additional bits compared to the other two methods. We can see that up to the value of m^* our proposed method matches (and in several instances it exceeds) the performance of $RHP(n + m)$, indicating that the estimation of the hash values for the m derived random vectors using the conditional probabilities is extremely accurate. We emphasize that for m^*=85 in Figure 3, our technique has practically the same performance as $RHP(n + m)$ even-though it uses only 32 bits per item compared to $32 + 85$=117 bits for the latter. The same trends appears in Figure 4 where our proposed methods is using only 12 bits per data item, while $RHP(n + m)$ needs at least 12+85=97 bits to achieve the same level of accuracy.

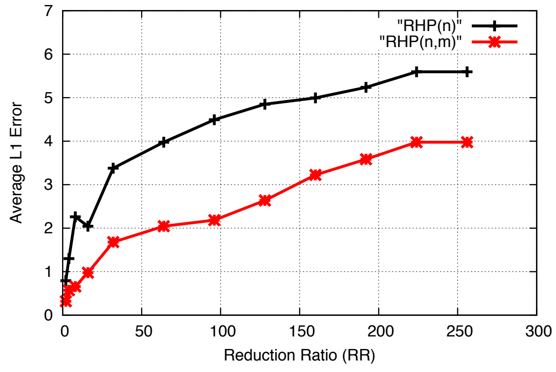

Figure 7: Intel Lab Data - TEMPERATURE

In Figures 5, 6, 7 and 8 we present graphs showing the accuracy of out $RHP(n, m)$ framework compared to the $RHP(n)$ technique for different reduction ratios (x-axis in the Figures). In all cases, our method provides significantly more accurate estimates, reducing the estimation error by up to 70%.

We further evaluate the extension to our framework described in Section 5.4. We trained our algorithms using a 10% sample of the NBA data set and then used a different selection of 100 data items for computing their similarity. During the evaluation process, we progressively replaced data items with randomly generated data points in R^{17}. These random points significantly differ from the sample that we used to train our method and present the worst case example for our technique as they uniformly cover the data space, negating any strong correlations between materialized and derived random vectors. On the contrary, they do not affect the basic $RHP(n)$ scheme that is oblivious to the data in query. The

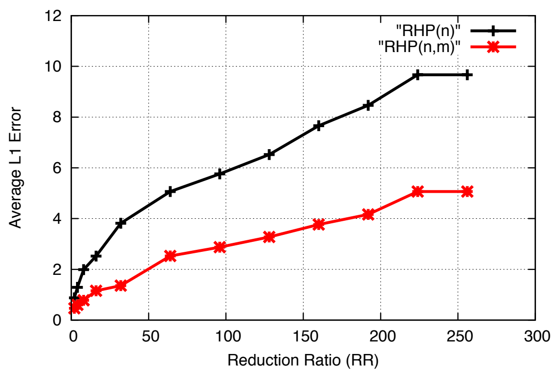

Figure 8: Intel Lab Data - HUMIDITY

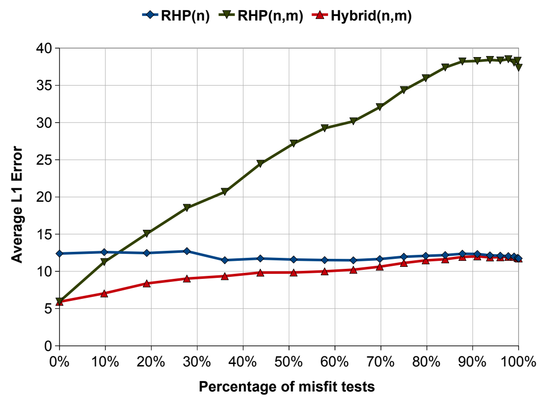

Figure 9: Performance when random data points are injected in the NBA data set

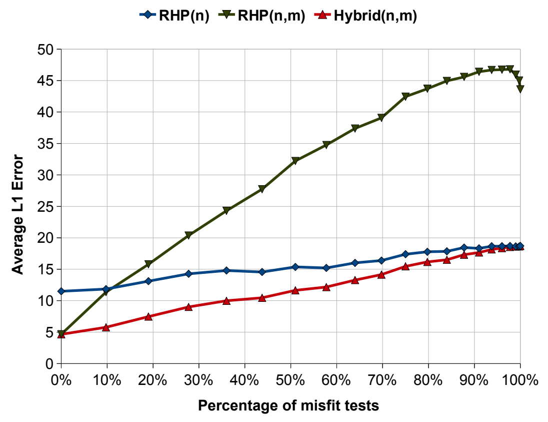

Figure 10: Performance when random data points are injected in the HOUSE data set

same procedure was repeated for the HOUSE data set using random points from R^6. In Figures 9, 10 we evaluate the performance of (i) our original framework, (ii) the classic $RHP(n)$ scheme, (iii) the extension described in Section 5.4, denoted as $Hybrid(n, m)$ in the graphs, for the NBA and HOUSE data sets respectively (θ_ϕ=30). Reduction ratio was 16:1 in all algorithms. We can see that our technique starts loosing its accuracy, when more random data items are used in the evaluation process (x-axis in the graphs depicts the percentage of tested pairs for which at least one of the data items was not from the original dataset). This is expected, as these random items do not follow the original data distribution. We can see in both data sets, that the Hybrid method retains the benefits of our technique and consistently outperforms the $RHP(n)$ scheme. This is because the Hybrid algorithm is far better than $RHP(n)$ for items selected from the original data (as is depicted in Figure 5, 6), while performs comparable to $RHP(n)$, when random data points are considered.

7. CONCLUSIONS

In this paper, we presented a LSH data reduction framework to perform similarity tests between data objects located at remote sites. Our method adopts the RHP framework but also takes into account the data distribution, detects correlations in the underlying data and achieves much better accuracy. We introduced a process for computing statistics from a RHP-projected data sample of higher dimensionality and then utilize these statistics in order to select a subset of possible dimensions on which the data is finally projected. We also discussed techniques that allow our algorithms to cope with changes in data distribution. A detailed experimental evaluation using several real data sets was presented and showed that our method outperforms the original RHP technique.

8. REFERENCES

[1] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[2] J. Chen, S. Kher, and A. Somani. Distributed Fault Detection of Wireless Sensor Networks. In *DIWANS*, 2006.

[3] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing Historical Information in Sensor Networks. In *ACM SIGMOD*, 2004.

[4] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical In-Network Data Aggregation with Quality Guarantees. In *Proceedings of EDBT*, 2004.

[5] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.

[6] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. TACO: Tunable Approximate Computation of Outliers in wireless sensor networks. In *SIGMOD*, 2010.

[7] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. QuickSAND: Quick Summary and Analysis of Network Data. Technical report, DIMACS 2001-43, Dec 2001.

[8] A. Gionis, D. Gunopulos, and N. Koudas. Efficient and tunable similar set retrieval. In *SIGMOD*, 2001.

[9] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6), 1995.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[11] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive*, 2006.

[12] N. Koudas, A. Marathe, and D. Srivastava. Propagating Updates in SPIDER. In *ICDE*, pages 1146–1153, 2007.

[13] D. Ravichandran, P. Pantel, and E. Hovy. Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In *ACL*, 2005.

[14] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis. SKYPEER: Efficient Subspace Skyline Computation over Distributed Data. In *Proceedings of ICDE*, 2007.

[15] X. Xiao, W. Peng, C. Hung, and W. Lee. Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks. In *MobiDE*, 2007.

[16] G. Xue, Y. Jiang, Y. You, and M. Li. A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme. In *UPGRADE*, 2007.