# SIMILARITY PRESERVING COMPRESSIONS OF HIGH DIMENSIONAL SPARSE DATA

**Raghav Kulkarni**
LinkedIn Bangalore, India
kulraghav@gmail.com

**Rameshwar Pratap**
IIIT Bangalore
rameshwar.pratap@gmail.com

## ABSTRACT

The rise of internet has resulted in an explosion of data consisting of millions of articles, images, songs, and videos. Most of this data is high dimensional and sparse, where the standard compression schemes, such as LSH (2; 4), become inefficient due to at least one of the following reasons: 1. Compression length is nearly linear in the dimension and grows inversely with the sparsity 2. Randomness used grows linearly with the product of dimension and compression length. We propose an efficient compression scheme mapping binary vectors into binary vectors and simultaneously preserving Hamming distance and Inner Product. Our schemes avoid all the above mentioned drawbacks for high dimensional sparse data. The length of our compression depends only on the sparsity and is independent of the dimension of the data, and our schemes work in the streaming setting as well. We generalize our scheme for real-valued data and obtain compressions for Euclidean distance, Inner Product, and $k$-way Inner Product.

## 1 INTRODUCTION

The technological advancements have led to the generation of huge amount of data over the web such as texts, images, audios, and videos. Needless to say that most of these datasets are high dimensional and sparse. Searching for similar data-objects in such massive and high dimensional datasets is becoming a fundamental subroutine in many scenarios like clustering, classification, nearest neighbors, ranking etc. However, due to the *"curse of dimensionality"* a brute-force way to compute the similarity scores on such data sets is very expensive and at times infeasible. Therefore it is quite natural to investigate the techniques that compress the dimension of dataset while preserving the similarity between data objects. There are various compressing schemes that have been already studied for different similarity measures. Below we discuss a few such notable schemes and their shortcoming in case of high dimensional sparse data. In this work we consider binary and real-valued datasets. For binary data we focus on Hamming distance and Inner product, while for real-valued data we focus on Euclidean distance and Inner product.

### 1.1 EXAMPLES OF SIMILARITY PRESERVING COMPRESSIONS AND THEIR SHORTCOMINGS

The quality of any compression scheme can be evaluated based on the following two parameters - 1) the *compression-length*, and 2) the amount of *randomness* required for the compression. The compression-length is defined as the dimension of the data after compression. Ideally, it is desirable to have both of these to be small while preserving a desired accuracy in the compression. Below we will notice that most of the above mentioned compression schemes become in-feasible in the case of high dimensional sparse datasets as 1) their compression-length is very high, and 2) the amount of randomness required for the compression is quite huge.

Gionis, Indyk, Motwani (2) proposed a data structure to solve approximate nearest neighbor ($c$-NN) problem in binary data for **Hamming distance**. Their scheme popularly known as **Locality Sensitive Hashing** (LSH). Intuitively, their data structure can be viewed as a compression of a binary vector, which is obtained by projecting it on a randomly chosen bit positions. A major disadvantage of their scheme is that their compression length (size of hash table) can be linear in the dimension.

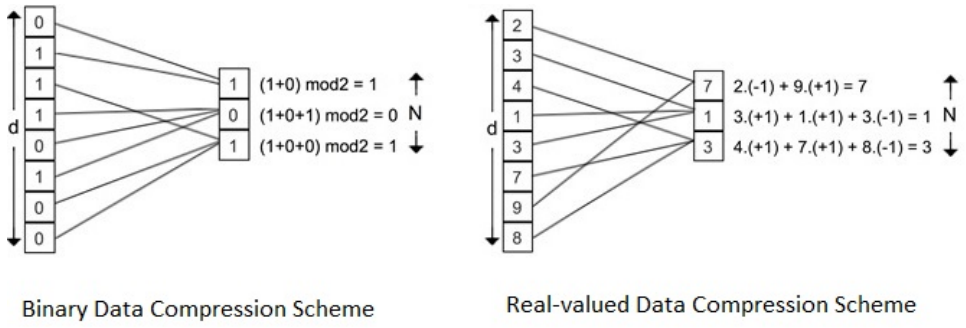
[0]A reference of the complete version of the paper is (7).

Ata Kabán (6) suggested a compression schemes for real data which preserves **inner product** *via* **random projection**. If the input data is binary, and it is desirable to get the compression only in binary, then to the best of our knowledge no such compression scheme is available. However, with some sparsity assumption, there are schemes (see (9)) available which *via asymmetric padding* reduce the inner product similarity to *Jaccard similarity*. Then *minhash permutations* – a compression scheme for Jaccard similarity, can be applied on the padded version of the data. A major disadvantage here is that for high dimensional data computing permutations are very expensive as the randomness required in this compression scheme is polynomial in the dimension.

**JL transform** (5) suggests a compressing scheme for real-valued data. For any $\epsilon > 0$, it compresses the dimension of the points from $d$ to $O\left(\frac{1}{\epsilon^2}\log n\right)$ while preserving the **Euclidean distance** between any pair of points within factor of $(1 \pm \epsilon)$. The compression-length in this scheme is $O\left(\frac{1}{\epsilon^2}\log n\right)$, and it requires $O\left(\frac{1}{\epsilon^2} d \log n\right)$ randomness.

## 2 Our contribution

In this work we present a compressing scheme for high dimensional sparse data. In contrast with the "local projection" strategies used by most of the previous schemes such as LSH (4; 2), JL transform (5), and (6) our scheme combines (using sparsity) the following two step approach 1. Partitioning the dimensions into several buckets, 2. Then obtaining "global linear summaries" of each of these buckets. Below, we pictorially describe an instance of our compression scheme – the left figure describe binary data compression scheme while the right figure is real-valued data compression scheme.




Binary Data Compression Scheme

Real-valued Data Compression Scheme

### 2.1 For binary data

For binary data, our compression scheme provides one-shot solution for both Hamming and Inner product – compressed data preserves both Hamming distance and Inner product. Moreover, the compression-length depends only on the sparsity of data and is independent of the dimension of data. We first define our compression scheme for binary data. In the results below, $\psi$ denote the maximum number of 1 in any vector; N denote the compression length; $\mathrm{d_H}(,)$ and $\mathrm{IP}(,)$ denote Hamming distance and inner product of two binary vectors, respectively.

**Definition 1 (Binary Compression Scheme)** *Let* N *be the number of buckets, for* $i = 1$ *to* $d$*, we randomly assign the* $i$*-th position to a bucket number* $b(i) \in \{1, \ldots \mathrm{N}\}$*. Then a vector* $\mathbf{u} \in \{0,1\}^d$*, compressed into a vector* $\mathbf{u'} \in \{0,1\}^{\mathrm{N}}$ *as follows:* $\mathbf{u'}[j] = \sum_{i:b(i)=j} \mathbf{u}[i] \pmod 2$.

**Theorem 1** *Consider a set* U *of binary vectors* $\{\mathbf{u_i}\}_{i=1}^n \subseteq \{0,1\}^d$*, a positive integer* $r$*, and* $\epsilon > 0$*. If* $\epsilon r > 3\log n$*, we set* $\mathrm{N} = O(\psi^2)$*; if* $\epsilon r < 3\log n$*, we set* $\mathrm{N} = O(\psi^2 \log^2 n)$*, and compress them into a set* $\mathrm{U'}$ *of binary vectors* $\{\mathbf{u'_i}\}_{i=1}^n \subseteq \{0,1\}^{\mathrm{N}}$ *using our Binary Compression Scheme. Then for all* $\mathbf{u_i}, \mathbf{u_j} \in \mathrm{U}$*,*

- *if* $\mathrm{d_H}(\mathbf{u_i}, \mathbf{u_j}) < r$*, then* $\Pr[\mathrm{d_H}(\mathbf{u_i}', \mathbf{u_j}') < r] = 1$*,*
- *if* $\mathrm{d_H}(\mathbf{u_i}, \mathbf{u_j}) \geq (1+\epsilon)r$*, then* $\Pr[\mathrm{d_H}(\mathbf{u_i}', \mathbf{u_j}') < r] < \frac{1}{n}$.

**Theorem 2** *Consider a set* U *of binary vectors* $\{\mathbf{u_i}\}_{i=1}^n \subseteq \{0,1\}^d$*, a positive integer* $r$*, and* $\epsilon > 0$*. If* $\epsilon r > 3\log n$*, we set* $\mathrm{N} = O(\psi^2)$*; if* $\epsilon r < 3\log n$*, we set* $\mathrm{N} = O(\psi^2 \log^2 n)$*, and compress them*

*into a set* $\mathrm{U}'$ *of binary vectors* $\{\mathbf{u_i}'\}_{i=1}^{n} \subseteq \{0,1\}^{\mathrm{N}}$ *using our Binary Compression Scheme. Then for all* $\mathbf{u_i}, \mathbf{u_j} \in \mathrm{U}$ *the following is true with probability at least* $1 - \frac{1}{n}$,

$$(1-\epsilon)\mathrm{IP}(\mathbf{u_i}, \mathbf{u_j}) \leq \mathrm{IP}(\mathbf{u_i}', \mathbf{u_j}') \leq (1+\epsilon)\mathrm{IP}(\mathbf{u_i}, \mathbf{u_j}).$$

In the following, we strengthen our result of Theorem 1, and shows a compression bound which is independent of the dimension and the sparsity, but depends only on the Hamming distance between the vectors. However, we could show our result in the expectation, and only for a pair of vectors.

**Theorem 3** *Consider two binary vectors* $\mathbf{u}, \mathbf{v} \in \{0,1\}^d$*, which get compressed into vectors* $\mathbf{u}', \mathbf{v}' \in \{0,1\}^{\mathrm{N}}$ *using our Binary Compression Scheme. If we set* $\mathrm{N} = O(r^2)$*, then*

- *if* $\mathrm{d_H}(\mathbf{u}, \mathbf{v}) < r$*, then* $\Pr[\mathrm{d_H}(\mathbf{u}', \mathbf{v}') < r] = 1$*, and*
- *if* $\mathrm{d_H}(\mathbf{u}, \mathbf{v}) \geq 4r$*, then* $\mathbb{E}[\mathrm{d_H}(\mathbf{u}', \mathbf{v}')] > 2r$.

**Remark 1** *For Hamming distance our scheme obtains the "no-false-negative" guarantee analogous to the one obtained in a recent paper by Pagh (8). When* $r$ *is constant, LSH (2) requires compression length linear in the dimension, however, due to Theorem 3, our compression length is only constant. Our compression length is* $O(\psi \log^2 n)$*, which is independent of the dimension* $d$*; whereas other schemes such as LSH may require the compression length growing linearly in* $d$*. Moreover, for all-pair compression for* $n$ *data points we use* $O(d(\log \psi + \log\log n))$ *randomness, which grows logarithmically in the sparsity and sub-logarithmically in terms of number of data points.*

### 2.2 For real-valued data

We generalize our scheme for real-valued data also and obtain compressions for Euclidean distance, Inner product, and $k$-way Inner product. In the below we state our result for euclidean distance, a similar result can also be obtained for inner product and $k$-way inner product.

**Definition 2 (Real-valued Compression Scheme)** *Let* $\mathrm{N}$ *be the number of buckets, for* $i = 1$ *to* $d$*, we randomly assign the* $i$*-th position to the bucket number* $b(i) \in \{1, \ldots \mathrm{N}\}$*. Then, for* $j = 1$ *to* $\mathrm{N}$*, the* $j$*-th coordinate of the compressed vector* $\boldsymbol{\alpha}$ *is computed as follows:* $\boldsymbol{\alpha}[j] = \sum_{i:b(i)=j} \mathbf{a}[i]x_i$*, where each* $x_i$ *is a random variable that takes a value between* $\{-1, +1\}$ *with probability* $1/2$.

**Theorem 4** *Consider two vectors* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$*, which get compressed into vectors* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^{\mathrm{N}}$ *using the compression scheme above. If we set* $\mathrm{N} = \frac{10\Psi^2}{\epsilon^2}$*, where* $\Psi = \max\{||\mathbf{a}||^2, ||\mathbf{b}||^2\}$ *and* $\epsilon > 0$*, then* $\Pr\left[|\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle - \langle \mathbf{a}, \mathbf{b} \rangle| > \epsilon\right] < 1/10$.

**Remark 2** *In order to compress a pair of data points our scheme requires* $O(d \log \mathrm{N})$ *randomness, which grows logarithmically in the compression length, whereas the other schemes require randomness which grows linearly in the compression length. Thus, when the number of points are small (constant), then for preserving a pairwise Inner product or Euclidean distance, we have a clear advantage on the amount of randomness required for the compression.*

## 3 Potential applications and open questions

A potential use of our result is to improve approximate nearest neighbor search via composing with LSH. One can first compress the input such that it preserve the desired similarity measure, and then can apply a collision based hashing algorithm such as LSH (2; 4) for efficient approximate nearest neighbor ($c$-NN) on the compressed data. There are many similarity based algorithmic methods used in large scale learning and information retrieval, e.g., Frequent itemset mining (1), ROCK clustering (3). One could potentially obtain algorithmic speed up in these methods via our compression schemes. Recently compression based on LSH for inner-product is used to speed up the forward and back-propagation in neural networks (10). One could potentially use our scheme to take advantage of sparsity and obtain further speed up.

Our work leaves the possibility of several open questions – improving the bounds of our compression scheme, and extending it to other similarity measures such as Cosine and Jaccard similarity are major open questions of our work.

## REFERENCES


[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[2] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 518–529, 1999.

[3] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Inf. Syst.*, 25(5):345–366, 2000.

[4] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613, 1998.

[5] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Conference in modern analysis and probability (New Haven, Conn., 1982), Amer. Math. Soc., Providence, R.I.*, pages 189–206, 1983.

[6] Ata Kaban. Improved bounds on the dot product under random projection and random sign projection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 487–496, 2015.

[7] Raghav Kulkarni and Rameshwar Pratap. Similarity preserving compressions of high dimensional sparse data. *CoRR*, abs/1612.06057, 2016.

[8] Rasmus Pagh. Locality-sensitive hashing without false negatives. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1–9, 2016.

[9] Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 981–991, 2015.

[10] Ryan Spring and Anshumali Shrivastava. Scalable and sustainable deep learning via randomized hashing. *CoRR*, abs/1602.08194, 2016.