Emmanouil Kritharakis

Implementing & Analyzing Johnson and Lindenstrauss Lemma Methods

Introduction

As a system Ph.D. student working on graph representation learning, I work daily with large graph-related datasets of billions of nodes and trillions of edges. Each of those nodes includes a multidimensional vector called a feature vector, which uniquely describes each inspected node respectively. The size of dimensions varies from a few hundred to thousands.

The lurking problems of high-dimensional data are numerous. First of all, the data could be more easily interpretable. In particular, as the number of dimensions increases, data cannot be visualized and understand its interconnections. Second, they are computationally expensive. In the systems field, resource utilization plays a significant role in performance. The system's memory footprint will be significant using a large number of dimensional data. Third, the curse of dimensionality affects the performance of the graph machine learning models. More specifically, the additional dimensions introduce noise to the model ushering lower accuracy metrics.

The research community has proposed some techniques for dimensionality reduction to resolve the above problems. First, the principle component analysis (PCA) is a linear method that transforms the data into a lower-dimensional space by finding the directions of maximum variance. Second, the Linear Discriminant Analysis (LDA) is a supervised method that finds the linear combinations of the features that maximize the separation between the classes. It reduces the dimensionality of the data by projecting it onto these combinations. Third, we have auto-encoders. They encode the data into a lower-dimensional representation and decode it back into the original space, minimizing the reconstruction error. Last but not least, there is the Johnson and Lindenstrauss Lemma, which we discussed during one of the lectures of this course. It will be analyzed in depth in the following sections.

This project aims to implement and analyze not only the vanilla Johnson and Lindernstrauss Lemma but also alternative methods such as the one discussed by Achlioptas [1] and Cohen, Jayram, & Nelson [2] publications. During the experimental section, I will conduct experiments comparing those methods based on their maximum, mean, and minimum distortion using both synthetic and real datasets. Also, we will systematically compare those approaches by their run time metrics. Moreover, we will investigate how tuning the sparsity matrix on Achlioptas paper affects the maximum distortion.

Vanilla Johnson and Lindenstrauss Lemma

As described in the Introduction section, the vanilla Johnson and Lindenstrauss Lemma is a technique to reduce the dimensions of input data. Moreover, it holds the identity that it preserves the pairwise distances between the data points or in other words it retains the geometric properties of the data. The lemma is the following: *For all $0 < \epsilon < 1$, integers $n, d > 1$ and $X \subset \mathbb{R}^d$ with $|X| = n$, there exists $f : X \to \mathbb{R}^m$ with $m = O(\epsilon^{-2} \cdot logn)$ such that:*

$$\forall y, z \in X, (1 - \epsilon) \cdot ||y - z||_2 \leq ||f(y) - f(z)||_2 \leq (1 + \epsilon) \cdot ||y - z||_2$$

The fig 1 visualize the lemma. X matrix represents the n vectors with d initial dimensions. Π matrix is an appropriately scaled orthogonal projection onto a uniformly random m-dimensional subspace. Last, the output data points has the target dimension M and the pairwise distances between the data points are maintained within an arbitrarily small factor ϵ.

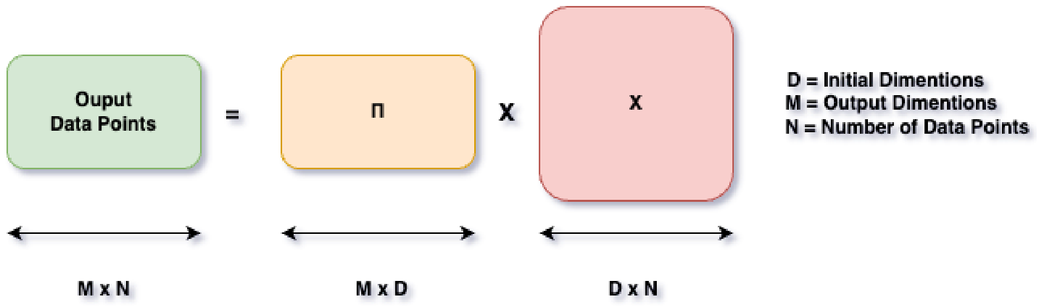


Figure 1: Visualization of Johnson and Lindenstrauss Lemma

Achlioptas Lemma Method

Achlioptas lemma method is an alternative to the original Johnson and Lindenstrauss lemma. In particular, all known constructions of such embeddings involve projecting the n points onto a random k-dimensional hyperplane. He gives a novel construction of the embedding, suitable for database applications, which amounts to computing a simple aggregate over k random attribute partitions. The proposed theorem is the following: *Let P be an arbitrary set of n points in $\mathbb{R}^d$ represented as an $n \times d$ matrix A. Given $\epsilon, \beta > 0$, let $k_0 = \frac{4+2\cdot\beta}{\frac{\epsilon^2}{2} - \frac{\epsilon^2}{3}} \cdot logn$. For integer $k \geq k_0$, let R be a $d \times k$ random matrix with $R(i,j) = r_{i,j}$, where $\{r_{i,j}\}$ are independent random variables from either one of the following two distributions:*

Case 0

$$r_{i,j} = \begin{cases} +1 & p = 1/2 \\ -1 & p = 1/2 \end{cases}$$

Case 1

$$r_{i,j} = \sqrt{3} \cdot \begin{cases} +1 & p = 1/6 \\ 0 & p = 2/3 \\ -1 & p = 1/6 \end{cases}$$

Let $E = \frac{1}{\sqrt{k}} \cdot A \cdot R$. A visualization of output matrix E is given in figure 2 Let $f : \mathbb{R}^d \to \mathbb{R}^k$ map the i^{th} row of A to the i^{th} row of E. With probability at least $1 - n^{-\beta}$, for all $u, v \in P$.

$$(1-\epsilon) \cdot ||u-v||_2 \leq ||f(u) - f(v)||_2 \leq (1+\epsilon) \cdot ||u-v||_2$$

Cohen, Jayram, & Nelson Lemma Method

Cohen, Jayram, & Nelson lemma method is an other alternative to the original Johnson and Lindenstrauss lemma. They propose that though the constructions given for default Π are simple, the analyses are not, employing intricate combinatorial arguments. They here give two simple alternative proofs of the main result of default Johnson and Lindenstrauss lemma, involving no delicate combinatorics. One of these proofs has

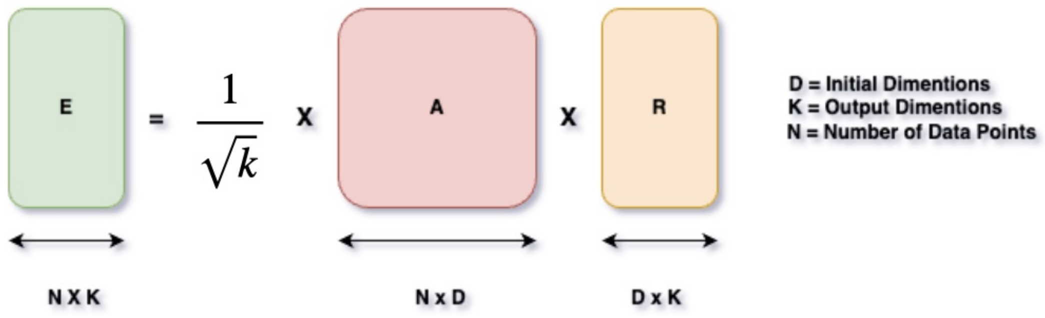
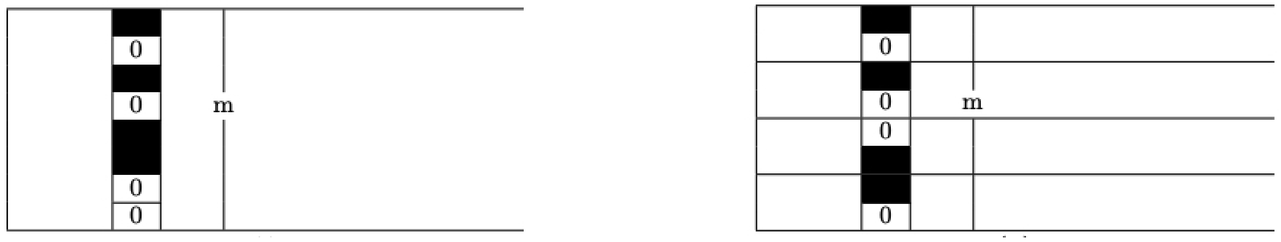


Figure 2: Visualization of Achlioptas Lemma Method

already been tested pedagogically, requiring slightly under forty minutes by the third author at a casual pace to cover all details in a blackboard course lecture arguments.

More specifically their JL distribution over Π has exactly s non-zero entries per column where s is radomly selected from $1 \leq s \leq m$. Each non-zero entry is a scaled Bernoulli-Rademacher. In particular, $\Pi \in \mathbb{R}^{m \times d}$ satisfies $\Pi_{i,j} = \frac{n_{r,i} \cdot \sigma_{r,i}}{\sqrt{s}}$, where $n_{r,i}$ are Bernoulli random variables and $\sigma_{r,i}$ are independent Rademachers. As Rademacher we define in this case a uniform random variable from $\{-1, 1\}$. The two provided cases are visualized in figure 3. Both cases are column based built. In particular, at case 0, each column has s non-zero entries with value equal to the above formula for $\Pi_{i,j}$, while at case 1, each column has again s non-zero entries but each column is split into $\frac{m}{s}$ blocks, in which only one random entry has a non-zero value.



(a) Case 0 (b) Case 1

Figure 3: Cohen, Jayram, & Nelson Lemma Proposed Π Matrix Cases

Experimental Section

At this section, I mention all the types of experiments I conducted including the parameter values I utilized. More specifically, I conducted 3 experiments. The first one is responsible to compare all the different approaches among them and measures the maximum, mean and minimum distortion of the euclidean distances between the data points. For this experiment, we used a synthetic dataset of 100 vectors with 10000 dimensions produced by a normal distribution. Figures 4, 5, & 6 present the results over the synthetic dataset we utilized. Those figures present the maximum, minimum and mean distortion over a range of target dimensions starting from 100 up to the initial 10000 for the input vectors. With red and green color the horizontal and vertical lines present the theoretical distortion and the theoretical target dimension the vanilla Johnson and Lindenstrauss lemma provides respectively. At this point, we should mention the total run time measurements for all the proposed approaches. As Table 1 shows vanilla and Achlioptas cases take approximately the same time, while the Cohen, Jayram & Nelson cases take more than 10x times more. I argue that the reason behind that is that Cohen, Jayram & Nelson cases built their matrices column wise by choosing randomly selected all the non-zero cells which takes more time in comparison to other methods. Of course, the total times are language depend, so given the fact that the implementation is based on Python and Python is not known for an efficient language the run time results might be faster with other languages such as Rust, C or C++.

For the second experiment, I conducted exactly the same process but this time I utilized a real dataset. The dataset is called OGB-products and the link for it can be found here. This dataset includes an undirected and unweighted graph, representing an Amazon product co-purchasing network. Specifically, node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100. For this experiment, I utilized 100 node feature vectors with 100 dimensions. The results are presented over the figures 7, 8, & 9.

For the third experiment, I researched the sparse density of Achlioptas lemma method. In particular, Achlioptas proposed two alternative cases for JL lemma. Both of them randomly selects matrix cell values with specific probability. The arisen question was how the final maximum distortion will be affected if we tune the probabilities given the fact that we introduce a δ parameter on them. More specifically, the two Achlioptas cases will transform as follows:

<u>Case 0</u>

$$r_{i,j} = \begin{cases} +1 & p = 1 - \delta \\ -1 & p = \delta \end{cases}$$

<u>Case 1</u>

$$r_{i,j} = \sqrt{3} \cdot \begin{cases} +1 & p = \frac{1-\delta}{2} \\ 0 & p = \delta \\ -1 & p = \frac{1+\delta}{2} \end{cases}$$

The experimental results are presented over the figures 10 & 11. In these figures, we locked the target dimension to 2000 and we managed to measure the maximum distortion by tuning the δ parameter.

Table 1: Run time table for all methods with synthetic dataset

JL Method	**Run Time(sec)**
Vanilla	**26.39**
Achlioptas Case 0	**32.99**
Achlioptas Case 1	**33.26**
Cohen Jayram, Nelson Case 0	**540.12**
Cohen Jayram, Nelson Case 1	**561.37**

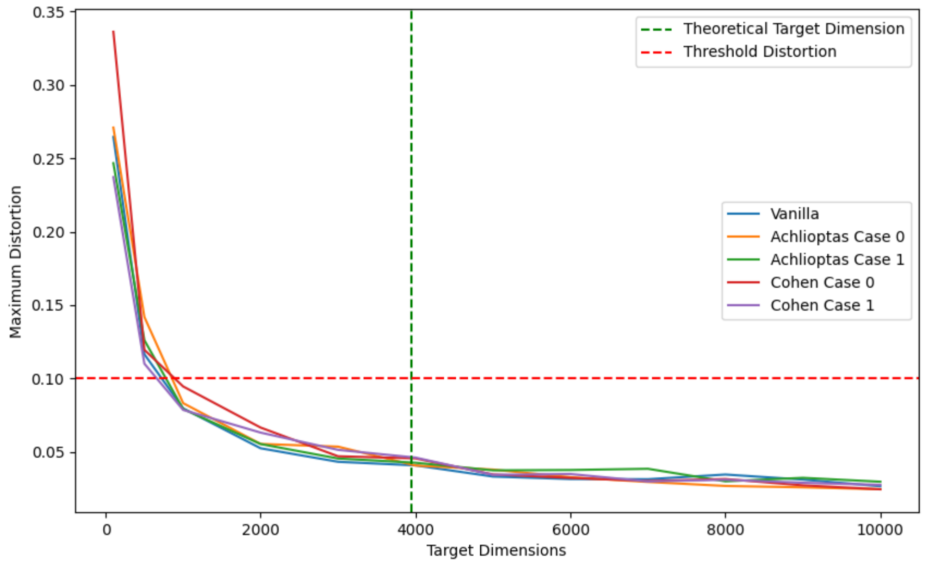


Figure 4: Maximum Distortion over different target dimensions with synthetic dataset. The green vertical line represents the theoretical target dimension from vanilla Johnson and Lindenstrauss lemma.

Last but not least, we should point out that our experiments were conducted 10 times each and the presented results are the average ones. The conclusions of those experiments are presented in the next section.

Conclusions

At this section, I mention all the project's milestones as well as the lessons learned through the process. Starting with the milestones:

- Implement the vanilla Johnson and Lindenstrauss lemma as well as 2 alternatives from related published work. In particular, both the Achlioptas's and the Cohen, Jayram, & Nelson's publications were implemented from scratch and tested experimentally over synthetic and real dataset (named OGB-Products).
- Both publications proposed 2 alternative cases which were tested over 3 metrics. More specifically, maximum, mean and minimum distortion between the euclidean distances among the data points were calculated and plotted.
- The total run time as well as the CPU utilization over all approaches were calculated and shared through table 1.
- Explore the effect of matrix sparsity over the Achlioptas solution experimentally.

In succession, I present the conclusions I draw through the project:

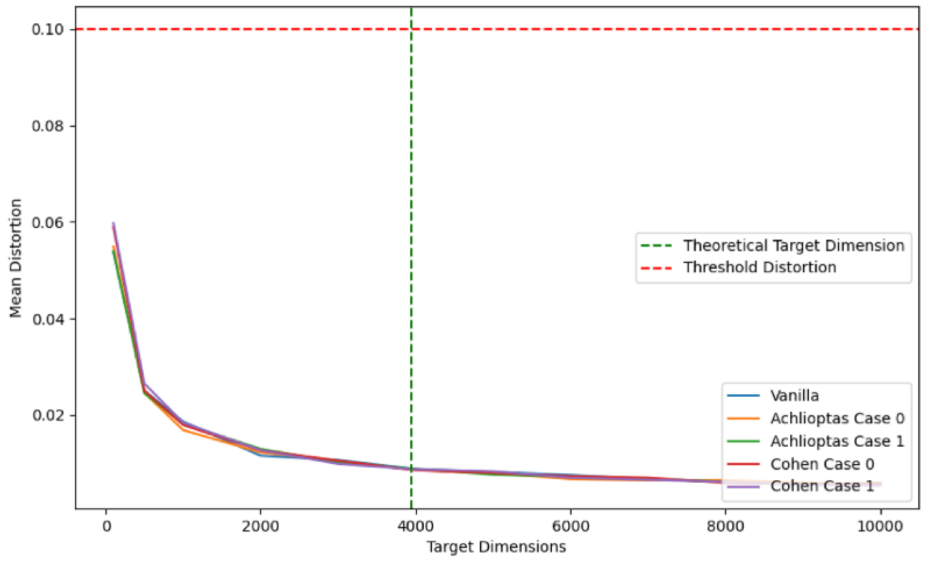


Figure 5: Mean Distortion over different target dimensions with synthetic dataset. The green vertical line represents the theoretical target dimension from vanilla Johnson and Lindenstrauss lemma.

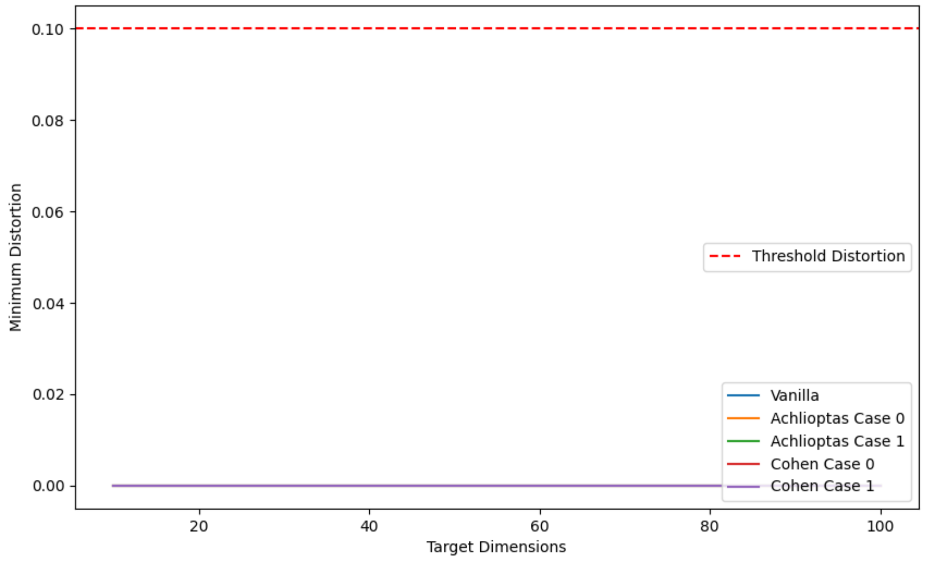


Figure 6: Minimum Distortion over different target dimensions with synthetic dataset. The green vertical line represents the theoretical target dimension from vanilla Johnson and Lindenstrauss lemma.

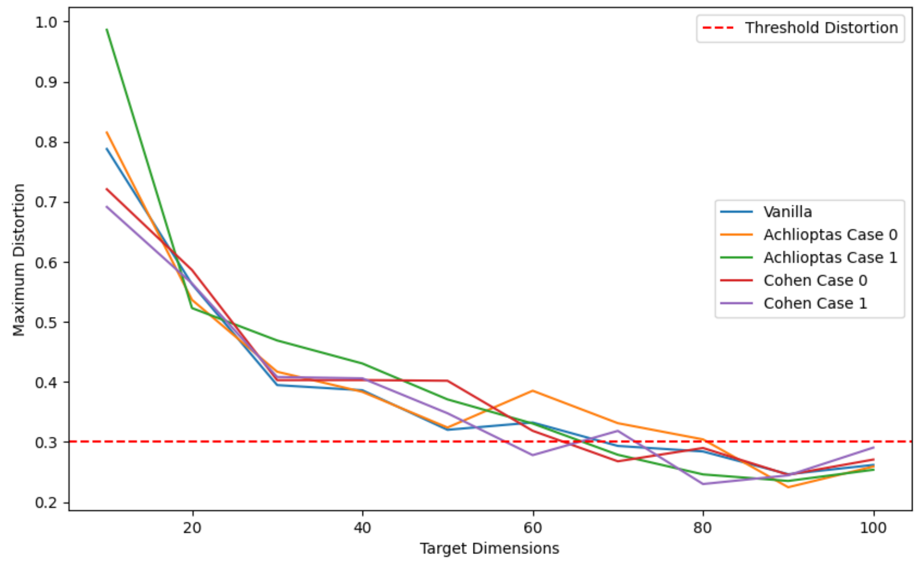


Figure 7: Maximum Distortion over different target dimensions with real dataset OGB-Products.

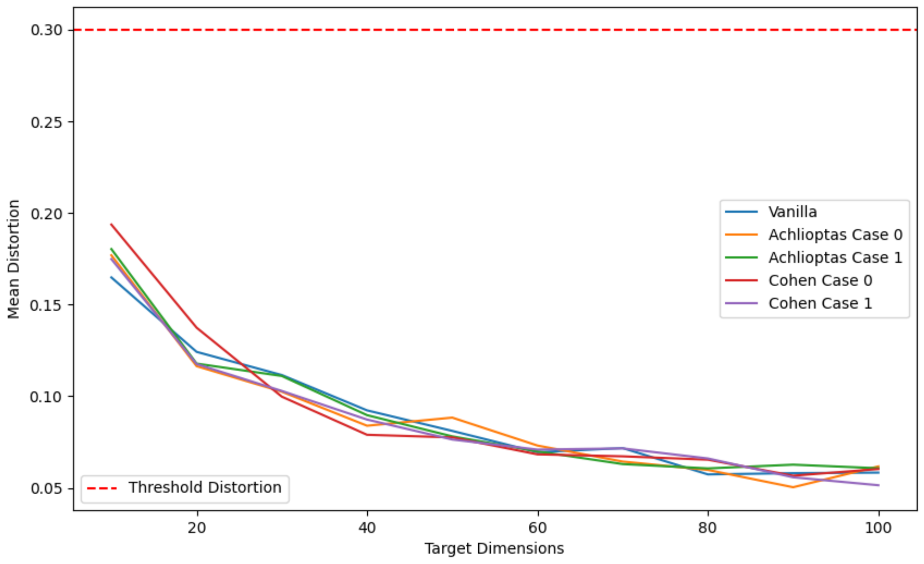


Figure 8: Mean Distortion over different target dimensions with real dataset OGB-Products.

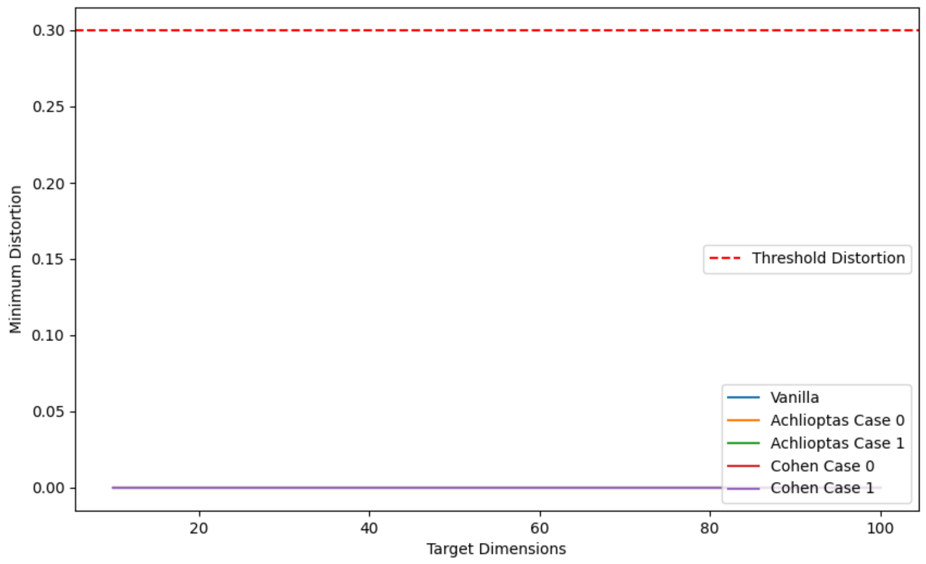


Figure 9: Minimum Distortion over different target dimensions with real dataset OGB-Products.

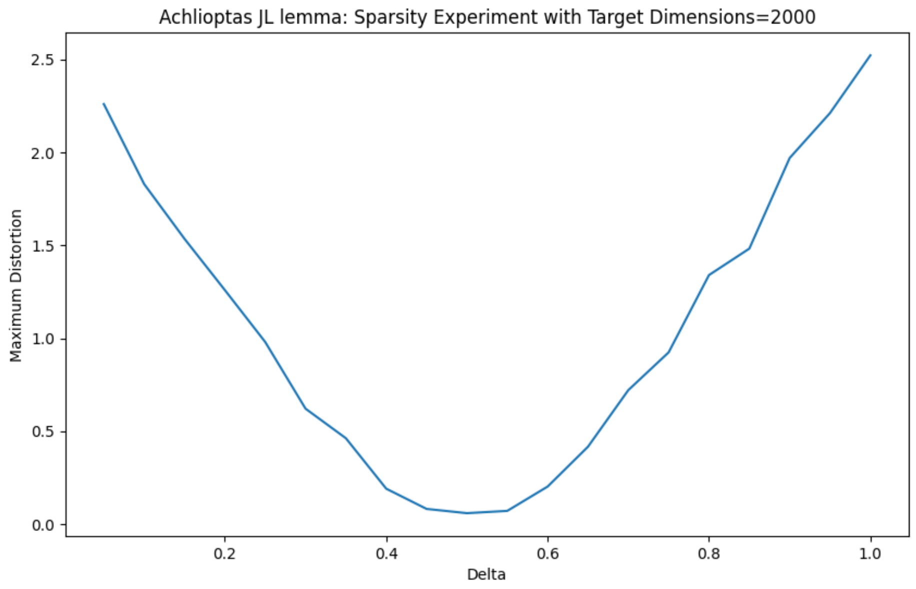


Figure 10: Explore Achlioptas case 0 by tuning parameter δ sparsity density and measuring maximum distortion over synthetic dataset.

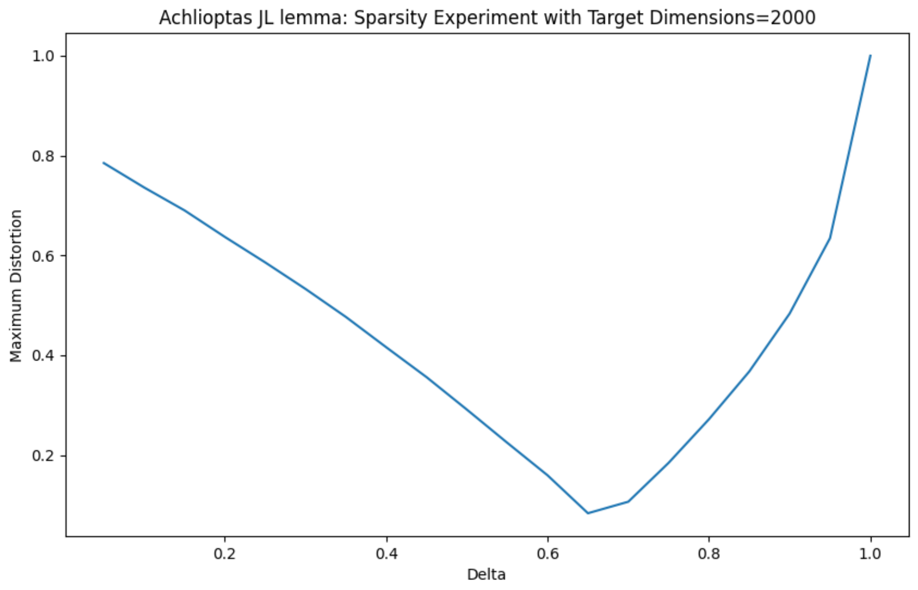


Figure 11: Explore Achlioptas case 1 by tuning parameter δ sparsity density and measuring maximum distortion over synthetic dataset.

- Vanilla, Achlioptas and Cohen, Jayram, & Nelson's lemma methods follow the same pattern in terms of maximum, mean and minimum distortion.
- The theoretical bound that vanilla Johnson and Lindenstrauss lemma provides is larger than the experimental one. The plots for maximum and mean distortion both for synthetic and real datasets indicate that the theoretical bound is loose and we can achieve the preservation of the euclidean distances among the data points with target dimensions less than the theoretical one.
- Both synthetic and real datasets respect the theoretical bound that the vanilla lemma provides.
- Prove experimentally that Achlioptas's choices over the probabilities of sparse matrices minimizes the total distortion.

Code Reproducibility

The Github repo is provided here. The code was written and tested on Ubuntu 20.04 operating system, with Python 3.9 installed. The only python packages required to be installed to run the code are the following: *numpy, pandas, matplotlib, sklearn, os*. To install them under pip you need to follow the command:

- pip install numpy pandas matplotlib sklearn os

The following lines provide you a description of how to run all the experiments of the project.

For synthetic dataset plots, run the *synthetic dataset* script as follows: **python synthetic_dataset.py**.

For real dataset plots, run the *real dataset* script as follows: **python real_dataset.py**.

For exploring Achlioptas plots, run the *explore Achlioptas* script as follows: **python explore_achlioptas.py**. Please be aware that the experiments take long (approximately over some hours) to be completed. Moreover, here I provide a link with the real dataset OGB-Products, which is not include in the Github repo due to its total size. If you want to run the real dataset experiment please download it and put the dataset folder under the project folder. All the plots will be generated under the figures folder inside sub folders with the proper naming for each experiment that was conducted.

References

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

[2] Michael B Cohen, TS Jayram, and Jelani Nelson. "Simple analyses of the sparse Johnson-Lindenstrauss transform". In: *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.