FURTHER TESTING AND COMPARISONS OF THE HYBRID-(L1, L2) SAMPLING ALGORITHM WITH L1 AND L2 SAMPLING

Yogesh Agrawal, Fernando Spadea

ABSTRACT

1 INTRODUCTION

Matrices are a very useful format for storing and working with data, but they can often become too large or dense to efficiently work with. Often, it is much more efficient to work with sparse matrices, or those with mostly 0 values, but most matrices are not going to be that clean. Thus, it's useful to be able to transform these matrices into sparse matrices while losing as little of the information they hold as possible. There exist several different algorithms to attempt to do just this. The main goal of these algorithms is generally to minimize the norm of the difference between the original and sparse matrix. This paper will look at Abhisek Kundu's algorithm (Kundu, 2015).

Principle Component Analysis (PCA) is also an extremely useful technique for extracting important information from data with many attributes. It can be costly to use every attribute of matrices with hundreds or even thousands of elements; that's where PCA comes in. It is a technique for creating a smaller set of variables that represent the significance of each data matrix. This allows algorithms to operate on large matrices without having to use each individual element of the matrices which can significantly speed up their run times. However, PCA is a process which generally requires access to the entire matrix, and that can be an obstacle in many cases. These cases could range to user recommendations where the recommendation algorithm doesn't have access to the user's entire history or simply a situation where data isn't being shared due to privacy concerns (Kundu, 2015). In these situations, one is left with little choice but to attempt to approximate the PCA features needed.

There exist algorithms to attempt to solve this problem. We begin to approach this problem by first looking at sampling algorithms for matrices. The two relevant to the work of this paper are l_1 and l_2 sampling (Achlioptas & McSherry, 2007). In sampling, we assign probabilities to each element of a matrix, and then use those probabilities to randomly select the desired number of elements to keep from said matrix. With $l1$ sampling, we assign these probabilities by taking the absolute value of an entry and dividing it by the 1-norm of the matrix (Achlioptas et al., 2013b). As for $l2$ sampling, we use the entry squared dividing by the 2-norm of the matrix instead Drineas & Zouzias (2011b). A proposed sparsification algorithm uses just l_1 sampling to create a sparser matrix (Arora et al., 2006). This is achieved by assigning elements with magnitude less than a threshold, t, to t, or $-t$ if the element is negative, with probability dictated by $l1$ sampling where it is set to 0 otherwise (Arora et al., 2006). However, Abhisek Kundu claims to have created an improved algorithm by taking advantage of both $l1$ and $l2$ sampling (Kundu, 2015). The author tested his algorithm with a few different data sets to compare it with just using $l1$ or $l2$ sampling (Kundu, 2015). Nevertheless, our paper aims to further test his algorithm against $l1$ and $l2$ sampling on several more distinct data sets.

The original paper tested their algorithm on Synthetic, TechTC, Handwritten Digit, and Stock data. However, the handwritten digit data testing was limited to 6, 9, and 1, so we will be testing the algorithm on the MNIST handwritten digit data set with digits 0-9 (Deng, 2012). We will also test the algorithm on a license plate dataset [1]. Lastly, we will test the algorithm on a data set of historical football match data [2]. These additional tests will help support or devalue the algorithm presented by Abhisek Kundu.

We will be looking to see what trade offs exist between the degree of sparsification and retaining of the data from the original matrix and how that ties into the PCA approximations. Even if this algorithm proves to be efficient, it is important to make sure that it actually does a good job of representing the original matrix or else it will fail.

[1] https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

[2] https://www.kaggle.com/datasets/tobycrabtree/nfl-scores-and-betting-data

2 INTRODUCTION TO HYBRID L1-L2 SAMPLING TECHNIQUE

The hybrid L1-L2 sampling technique for matrix sparsification is an approach that combines the benefits of L1 sampling and L2 sampling to achieve a more efficient and accurate sparsification process. Matrix sparsification is the process of reducing the number of non-zero elements in a matrix while preserving its essential properties, such as spectral or structural information. This is particularly useful in large-scale data analysis, machine learning, and numerical optimization, where working with dense matrices can be computationally expensive.

L1 sampling is a technique that selects a small subset of rows or columns from the input matrix by considering their L1-norms (the sum of absolute values of their entries). Rows or columns with higher L1-norms are more likely to be selected, as they typically contain more information. L1 sampling can be effective in preserving the structure of the input matrix but may have difficulty preserving spectral properties.

L2 sampling, on the other hand, considers the L2-norms of the rows or columns (the square root of the sum of the squares of their entries). Rows or columns with higher L2-norms are more likely to be selected, as they often contain more information in terms of spectral properties. L2 sampling is better at preserving spectral properties but may struggle to preserve the structural properties of the input matrix.

The hybrid L1-L2 sampling technique combines these two approaches by taking into account both L1 and L2-norms. This is done by calculating a combined score for each row or column based on a weighted combination of their L1 and L2-norms. The weights can be adjusted to favor either L1 or L2-norms, depending on the desired balance between preserving structural and spectral properties.

In summary, the hybrid L1-L2 sampling technique for matrix sparsification aims to achieve a better balance between preserving the structural and spectral properties of the input matrix by combining the strengths of L1 and L2 sampling techniques. This can result in more accurate and efficient sparsification, which can be particularly beneficial in large-scale data analysis, machine learning, and numerical optimization applications.

3 ADVANTAGES AND DISADVANTAGES

The hybrid L1-L2 sampling technique for matrix sparsification comes with its own set of advantages and disadvantages.

Advantages:

1. Balanced sparsification: The hybrid approach allows for a more balanced sparsification by considering both the L1 and L2-norms of the rows or columns. This can result in better preservation of both structural and spectral properties of the input matrix.
2. Flexibility: The method offers flexibility in terms of the weights assigned to the L1 and L2-norms. Depending on the specific problem or application, the weights can be adjusted to favor either L1 or L2-norms, allowing the user to emphasize either structural or spectral properties as needed.
3. Improved accuracy: By considering both L1 and L2-norms, the hybrid approach can often achieve more accurate sparsification compared to using either L1 or L2 sampling alone. This can lead to better performance in downstream tasks, such as data analysis, machine learning, and numerical optimization.
4. Scalability: The hybrid L1-L2 sampling technique is applicable to large-scale matrices, making it suitable for problems where computational resources are limited or where working with dense matrices is computationally expensive.

Disadvantages:

1. complexity: The hybrid approach requires the calculation of both L1 and L2-norms, as well as the assignment of weights for their combination. This can result in increased complexity compared to using either L1 or L2 sampling alone.
2. Tuning parameters: Determining the appropriate weights for combining the L1 and L2-norms may require additional tuning or experimentation, which can be time-consuming and may not always guarantee optimal results.
3. Loss of sparsity: Although the hybrid approach aims to preserve the essential properties of the input matrix, it may still result in a loss of sparsity. This can be an issue if the primary goal is to minimize the number of non-zero elements in the resulting sparse matrix.

4. No universal solution: The hybrid L1-L2 sampling technique may not be the best solution for all problems. Depending on the specific properties of the input matrix and the goals of the sparsification process, other techniques or specialized algorithms might be more appropriate.

In conclusion, the hybrid L1-L2 sampling technique offers a flexible and balanced approach to matrix sparsification, but it comes with some trade-offs, such as increased complexity and the need for parameter tuning. The suitability of this approach depends on the specific problem, application, and desired balance between preserving structural and spectral properties.

4 POSSIBLE STRUGGLES

The hybrid L1-L2 sampling technique for matrix sparsification might struggle with certain types of data inputs or problem characteristics:

1. Highly imbalanced data: If the input matrix has extreme differences in the magnitudes of its elements or in the distribution of its non-zero elements, the hybrid approach may struggle to find an appropriate balance between preserving structural and spectral properties. In such cases, the method might require extensive tuning of the weights assigned to L1 and L2-norms.
2. Noise-dominated data: In the presence of significant noise in the input matrix, the hybrid L1-L2 sampling technique may have difficulty distinguishing between informative elements and noise, which could affect the accuracy of the sparsification process.
3. Strongly structured matrices: If the input matrix exhibits strong structural patterns or specific properties, such as block structure, banded structure, or specific types of rank structure, the hybrid L1-L2 sampling technique may not be the most suitable method. In these cases, specialized sparsification techniques tailored to the specific matrix structure might yield better results.
4. Highly sparse matrices: For matrices that are already very sparse, the hybrid L1-L2 sampling technique might not provide significant benefits over other methods or even the original sparse matrix. In these cases, the added complexity of the hybrid approach may not be justified.
5. Data with specific requirements: In some applications, it may be necessary to prioritize either structural or spectral properties of the input matrix to a greater extent than the hybrid L1-L2 sampling technique allows. In such cases, using L1 or L2 sampling alone or employing other specialized sparsification methods might be more appropriate.

While the hybrid L1-L2 sampling technique can be an effective method for matrix sparsification in many cases, it is not a one-size-fits-all solution. The suitability of this approach for a given problem depends on the characteristics of the input matrix and the specific goals of the sparsification process.

4.1 EXAMPLE

Suppose we have a large block-diagonal matrix, where the matrix is composed of smaller, dense submatrices along the main diagonal, and the off-diagonal elements are zero. Such a matrix can arise in applications like multi-domain physics simulations or multi-block systems.In this case, the hybrid L1-L2 sampling technique might be inefficient due to the following reasons:

1. The matrix structure is already sparse: Since the off-diagonal blocks of the matrix are filled with zeros, the matrix is already sparse. Applying the hybrid L1-L2 sampling technique might not provide significant benefits in terms of sparsity or computational savings.
2. Loss of block structure: The hybrid L1-L2 sampling technique does not specifically take into account the block structure of the input matrix. By sampling rows and columns based on their L1 and L2-norms, the resulting sparse matrix might lose the original block structure, which could be important for downstream applications.
3. Inefficient sampling: The L1 and L2-norms might not provide meaningful information for selecting rows and columns in this case. Since the matrix is already block-diagonal, it would be more efficient to focus on preserving the structure of the diagonal blocks themselves, rather than considering the norms of the rows and columns across the entire matrix.

In this example, it might be more appropriate to use specialized sparsification techniques tailored to the block structure of the matrix, such as block-wise sparsification or hierarchical sampling methods. These methods can better preserve the block structure and potentially provide more accurate and efficient sparsification.

5 METHODOLOGY AND KEY FORMULAE

Let us define the sampling operator $S_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ which will extract the elements from our given matrix $\mathbf{A}$. Here Ω is defined as multi-set of sampled indices (i_t, j_t) where t is an integer between 1 and s. Then:

$$S_\Omega = \frac{1}{s}\sum_{t=1}^{s} \frac{\mathbf{A}_{i_t j_t}}{p_{i_t j_t}} \boldsymbol{e}_{i_t} \boldsymbol{e}_{j_t}^T \quad \text{where}(i_t, j_t) \in \Omega$$

To sample s elements, we will use algorithm given in Algorithm 1 based on the probability distribution $\{p_{ij}\}_{i,j=1}^{m,n}$ which we calculate using the formula:

$$p_{ij} = \alpha\frac{|\mathbf{A}_{ij}|}{||\mathbf{A}||_1} + (1-\alpha)\frac{\mathbf{A}_i j^2}{||\mathbf{A}||_F^2||^2} \quad \text{where}\alpha \in (0,1] \tag{1}$$

Lemma of matrix-Bernstein gives proof for the quality of the approximation. The theorem goes as follows:
Theorem 1 Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\epsilon > 0$ be an accuracy parameter. Let $\mathbf{S}_\Omega$ be the sampling operator defined above and assume we generate multi-set Ω using the above sampling probabilities. Then, with probability at least $1-\delta$,

$$||\mathbf{S}_\Omega(\mathbf{A}) - \mathbf{A}||_2 \le \epsilon||\mathbf{A}||_2 \tag{2}$$

if

$$s \ge \frac{2}{\epsilon^2||\mathbf{A}||_2^2}\left(\rho^2(\alpha) + \gamma(\alpha) \in ||\mathbf{A}||_2/3\right)\ln\left(\frac{m+n}{\delta}\right) \tag{3}$$

where

$$\xi_{ij} = ||\mathbf{A}||_F^2 / \left(\frac{\alpha * ||\mathbf{A}||_F^2}{|\mathbf{A}_{ij}| * ||\mathbf{A}||_1} + (1-\alpha)\right) \quad \text{for}\mathbf{A}_{ij} \neq 0$$

$$\rho^2(\alpha) = \max\left(\max_i \sum_{j=1}^{n} \xi_{ij}\right) - \sigma_{min}^2(\mathbf{A})$$

$$\gamma(\alpha) = \max_{i,j:\mathbf{A}_{ij}\neq 0}\left(\frac{||\mathbf{A}||_1}{\alpha + (1-\alpha)\frac{||\mathbf{A}||_1 * |\mathbf{A}_{ij}|}{||\mathbf{A}||_F^2}}\right) + ||\mathbf{A}||_2$$

where σ_{min} refers to the smallest singular value of $\mathbf{A}$. Then, to find the optimal α^* and s^* by solving following optimization problem:

$$\alpha^* = \min_{\alpha\in(0,1]} f(\alpha) = \rho^2(\alpha) + \gamma(\alpha)\epsilon||\mathbf{A}||_2/3 \tag{4}$$

$$s^* = \frac{2}{\epsilon^2||\mathbf{A}||_2^2}\left(\rho^2(\alpha^*) + \gamma(\alpha^*) \in ||\mathbf{A}||_2/3\right)\ln\left(\frac{m+n}{\delta}\right) \tag{5}$$

Here, Matrix-Bernstein inequality gave us the functional norm in (3). This gives us a way to calculate the optimum α for our hybrid sampling.

For the **One Pass Hybrid** L_1, L_2 **Sampling**, we use **Algorithm 3** to get an iterative estimate of α^* in one pass over $\mathbf{A}$. For that, we need additional independent multisets $\mathbf{S}_3$ and $\mathbf{S}_4$ to learn the parameter α^*. While **Algorithm 2** requires double the memory required by $\boldsymbol{l}_1$ or $\boldsymbol{l}_2$ sampling. For **Algorithm 3** we need four times the memory. However, the asymptotic memory required in both remains $\mathcal{O}(s)$. To obtain the required independent random multiset of triples $\mathbf{S}_3$ and $\mathbf{S}_4$ each containing $\boldsymbol{s}$ elements from $\mathbf{A}$ in one pass. We can create sparse random matrix as shown in **Algorithm 3** that is unbiased estimator of $\mathbf{A}$. Then, we can create proxy for $\mathbf{A}$ to estimate the quantities needed to solve following optimization problem:

$$\tilde{\alpha} : \min_{\alpha\in(0,1]}\left(\tilde{\rho^2}(\alpha) + \tilde{\gamma}(\alpha) \in ||\mathbf{X}||_2/3\right) \tag{6}$$

where, for all$(i,j) \in S(:, 1:2)$:

$$\tilde{\xi_{ij}} = \frac{||X||_2^F}{\alpha\cdot||X||_2^F|X_{ij}|\cdot||X||_1 + (1-\alpha)} \tag{7}$$

$$r\tilde{h}o_2(\alpha) = \max\left(\max_i \sum_{j=1}^{n} \tilde{\xi}_{ij}, \max_j \sum_{i=1}^{m} \tilde{\xi}_{ij}\right) \tag{8}$$

$$\tilde{\gamma}(\alpha) = \max_{ij}\left\{\frac{||X||_1}{\alpha + (1-\alpha)\frac{||X||_1 \cdot |X_{ij}|}{||X||_2^F}}\right\} + ||X||_F \tag{9}$$

We note that $||X||_0 \leq s$. We can compute the quantities $\tilde{\rho}(\alpha)$ and $\tilde{\gamma}(\alpha)$, for a fixed α, using $O(s)$ memory. We consider $\varepsilon = \epsilon \cdot ||X||_2$ to be the given accuracy.

Theorem 3 shows the quality of approximation of principal components produced by **Algorithm 4**.

Theorem 3 Let $A \in \mathbb{R}^{m\times n}$ be a given matrix, and $\tilde{A}$ be a sparse sketch produced by Algorithm 1. Let $\tilde{V}_k$ be the PCA's of $\tilde{A}$ computed in step 3 of Algorithm 4. Then

$$\|A - A\tilde{V}_k\tilde{V}_k^T\|_F^2 \leq \|A - A_k\|_F^2 + \frac{4\|A\|_F^2}{\sigma_k(A)}\left\|A - \tilde{A}\right\|_2^2,$$

$$\|A_k - \tilde{A}_k\|_F \leq \sqrt{8k \cdot \left(\|A - A_k\|_2 + \left\|A - \tilde{A}\right\|_2^2\right)},$$

$$\|A - \tilde{A}_k\|_F \leq \|A - A_k\|_F + \sqrt{8k \cdot \left(\|A - A_k\|_2 + \left\|A - \tilde{A}\right\|_2^2\right)}.$$

The first inequality of Theorem 3 bounds the approximation of projected data onto the space spanned by top k approximate PCA's. The second and third inequalities measure the quality of $\tilde{A}_k$ as a surrogate for A_k and the quality of projection of sparsified data onto approximate PCA's, respectively.

6 ALGORITHMS

The algorithms used in this paper are brought over directly from Kundu's paper. There are four separate algorithms are that are used in combination to approximate PCA with limited data.

Algorithm 1 Element-wise Matrix Sparsification

Require: $A \in \mathbb{R}^{m\times n}$, accuracy parameter $\epsilon > 0$.

1: Set s as in eq. 5.
2: **for** $t = 1 \ldots s$ (i.i.d. trials with replacement) **do**
3: Randomly sample pairs of indices $(i_t, j_t) \in [m] \times [n]$ with $P[(i_t, j_t) = (i, j)] = p_{ij}$, where p_{ij} are as in 1, using α as in 4.
4: **end for**
5: **Output(sparse)**: $S_\Omega(A) = \frac{1}{s}\sum_{t=1}^{s} \frac{A_{i_t j_t}}{p_{i_t j_t}} e_{i_t} e_{j_t}^T$.

Algorithm 2 One-pass hybrid-(ℓ_1, ℓ_2) sampling

Require: A_{ij} for all $(i,j) \in [m] \times [n]$, arbitrarily ordered, and sample size s.

Apply SELECT algorithm in parallel with $O(s)$ memory using ℓ_1 probabilities to sample s independent indices (i_{t_1}, j_{t_1}) and corresponding elements $A_{i_{t_1} j_{t_1}}$ to form random multiset S_1 of triples $(i_{t_1}, j_{t_1}, A_{i_{t_1} j_{t_1}})$, for $t_1 = 1, \ldots, s$.

Run step 2 in parallel to form another independent multiset S_3 of triples $(i_{t_3}, j_{t_3}, A_{i_{t_3} j_{t_3}})$, for $t_3 = 1, \ldots, s$. (This step is only for Algorithm 3)

Apply SELECT algorithm in parallel with $O(s)$ memory using ℓ_2 probabilities to sample s independent indices (i_{t_2}, j_{t_2}) and corresponding elements $A_{i_{t_2} j_{t_2}}$ to form random multiset S_2 of triples $(i_{t_2}, j_{t_2}, A_{i_{t_2} j_{t_2}})$, for $t_2 = 1, \ldots, s$.

Run step 4 in parallel to form another independent multiset S_4 of triples $(i_{t_4}, j_{t_4}, A_{i_{t_4} j_{t_4}})$, for $t_4 = 1, \ldots, s$. (This step is only for Algorithm 3)

Compute and store $\|A\|_F^2$ and $\|A\|_1$ in parallel.

Set the value of $\alpha \in (0, 1]$ (using Algorithm 3).

Create empty multiset of triples S.

$X \leftarrow 0^{m \times n}$.

for $t = 1 \ldots s$ **do**

  Generate a uniform random number $x \in [0, 1]$.

  if $x \geq \alpha$ **then**

    $S(t) \leftarrow S_1(t)$;

  else

    $S(t) \leftarrow S_2(t)$.

  end if

  $(i_t, j_t) \leftarrow S(t, 1:2)$.

  $p \leftarrow \alpha \cdot \frac{|S(t,3)|}{\|A\|_1} + (1 - \alpha) \cdot \frac{|S(t,3)|^2}{\|A\|_F^2}$

  $X \leftarrow X + \frac{S(t,3)}{p \cdot s} e_{i_t} e_{j_t}^T$.

end for

Output: random multiset S, and sparse matrix X.

Algorithm 3 Iterative estimate of α^*

Require: Multiset of triples S_3 and S_4 with s elements each, number of iterations τ, accuracy ϵ, $\|A\|_F^2$, and $\|A\|_1$.

Create empty multiset of triples S.

$\alpha_0 = 0.5$

for $k = 1 \ldots \tau$ **do**

  $X \leftarrow 0^{m \times n}$.

  for $t = 1 \ldots s$ **do**

    Generate a uniform random number $x \in [0, 1]$.

    if $x \geq \alpha_{k-1}$ **then**

      $S(t) \leftarrow S_3(t)$;

    else

      $S(t) \leftarrow S_4(t)$.

    end if

    $(i_t, j_t) \leftarrow S(t, 1:2)$.

    $p \leftarrow \alpha_{k-1} \cdot \frac{|S(t,3)|}{\|A\|_1} + (1 - \alpha_{k-1}) \cdot \frac{|S(t,3)|^2}{\|A\|_F^2}$

    $X \leftarrow X + \frac{S(t,3)}{p \cdot s} e_{i_t} e_{j_t}^T$.

  end for

  $\alpha_k \leftarrow \tilde{\alpha}$ in 6 using X.

end for

Output: α_τ.

Algorithm 4 Fast Approximation of PCA

Require: Centered data $A \in \mathbb{R}^{m\times n}$, sparsity parameter $s > 0$, and rank parameter k.
1: Produce sparse unbiased estimator $\tilde{A}$ from A, in s i.i.d. trials using Algorithm 1.
2: Perform rank truncated SVD on sparse matrix $\tilde{A}$, i.e., $[\tilde{U}_k, \tilde{D}_k, \tilde{V}_k] = \text{SVD}(\tilde{A}, k)$.
3: **Output**: $\tilde{V}_k$ (columns of $\tilde{V}_k$ are the ordered PCA's).

7 PROOF

The proofs in this paper are brought directly from Kundus' paper.

7.1 THEOREM 1

To prove the above mentioned **Theorem 1** by following the method used by Kundu in their paper. We use the non-commutative matrix valued Bernstein bound of Recht (2011) and the proof outline of Achlioptas et al. (2013a) and Drineas & Zouzias (2011a). Rephrasing the matrix Bernstein bound as per our notations, we get:
Lemma 1 [Theorem 3.2 of Recht (2011)]
Let $M_1, M_2, \ldots, M_s$ be independent, zero-mean random matrices in $\mathbb{R}^{m\times n}$. Suppose

$$\max_{t\in[s]} \left\{ \mathbb{E}(M_t M_t^T)^2, \mathbb{E}(M_t^T M_t)^2 \right\} \leq \rho^2$$

and $\|M_t\|_2 \leq \gamma$ for all $t \in [s]$. Then, for any $\varepsilon > 0$,

$$\left\| \frac{1}{s} \sum_{t=1}^{s} M_t \right\|_2 \leq \varepsilon$$

holds, subject to a failure probability at most

$$(m+n) \exp\left(-\frac{s\varepsilon^2}{2\rho^2 + \frac{2}{3}\gamma\varepsilon} \right).$$

For all $t \in [s]$ we define the matrix $M_t \in \mathbb{R}^{m\times n}$ as follows:

$$M_t = \frac{A_{it} j_t}{p_{it} j_t} e_{it} e_j^T - A.$$

It now follows that

$$\frac{1}{s} \sum_{t=1}^{s} M_t = \frac{1}{s} \sum_{t=1}^{s} \left(\frac{A_{it} j_t}{p_{it} j_t} e_{it} e_j^T - A \right) = S\Omega(A) - A.$$

We can bound $\|M_t\|_2$ for all $t \in [s]$. We define the following quantity:

$$\lambda = \frac{\|A\|_1 \cdot |A_{ij}|}{\|A\|_F^2}, \quad \text{for } A_{ij} \neq 0.$$

Lemma 2 Using our notation, and using probabilities from the formula 1, for all $t \in [s]$,

$$\|M_t\|_2 \leq \max_{i,j:A_{ij}\neq 0} \left\{ \|A\|_1 \frac{\alpha + (1-\alpha)\lambda}{\|A\|_F^2} + \|A\|_2 \right\}.$$

Proof: Using probabilities of the form (1.2), and because $A_{ij} = 0$ is never sampled,

$$\|M_t\|_2 = \left\| \frac{A_{it} j_t}{p_{it} j_t} e_{it} e_j^T - A \right\|_2 \leq \max_{i,j:A_{ij}\neq 0} \left\{ \frac{\alpha\|A\|_1 + (1-\alpha)|A_{ij}|}{\|A\|_F^2} \right\} + \|A\|_2$$

Using (1.8), we obtain the bound. □

Next, we bound the spectral norm of the expectation of $M_t M_t^T$.

Lemma 3 Using our notation, and using probabilities of the form (1.2), for all $t \in [s]$,

$$\mathbb{E}(M_t M_t^T)^2 \leq \|A\|_F^2 \frac{\beta_1 - \sigma^2 \min(A)}{\|A\|_F^2},$$

where,

$$\beta_1 = \max_i \sum_{j=1}^{n} \left\{ \frac{\alpha \|A\|_F^2}{|A_{ij}| \|A\|_1} + (1-\alpha) \right\}^{-1}, \quad \text{for } A_{ij} \neq 0.$$

Proof: Recall that $A = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} e_i e_j^T$ and $M_t = \frac{A_{it j_t}}{p_{it j_t}} e_{it} e_j^T - A$ to derive

$$\begin{aligned}
\mathbb{E}[M_t M_t^T] &= \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} \left(\frac{A_{ij}}{p_{ij}} e_i e_j^T - A \right) \left(\frac{A_{ij}}{p_{ij}} e_j e_i^T - A^T \right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{A_{ij}^2}{p_{ij}} e_i e_i^T - A A^T.
\end{aligned}$$

Sampling according to probabilities of eqn. (1.2), and because $A_{ij} = 0$ is never sampled, we get, for $A_{ij} \neq 0$,

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=1}^{n} \frac{A_{ij}^2}{p_{ij}} &= \|A\|_F^2 \sum_{i=1}^{m} \sum_{j=1}^{n} \left\{ \frac{\alpha \|A\|_F^2}{|A_{ij}| \|A\|_1} + (1-\alpha) \right\}^{-1}, \\
&\leq \|A\|_F^2 \sum_{i=1}^{m} \max_i \sum_{j=1}^{n} \left\{ \frac{\alpha \|A\|_F^2}{|A_{ij}| \|A\|_1} + (1-\alpha) \right\}^{-1}.
\end{aligned}$$

Thus,

$$\mathbb{E}[M_t M_t^T] \preceq \|A\|_F^2 \beta_1 \sum_{i=1}^{m} e_i e_i^T - A A^T = \|A\|_F^2 \beta_1 I_m - A A^T.$$

Note that, $\|A\|_F^2 \beta_1 I_m$ is a diagonal matrix with all entries non-negative, and AA^T is a positive semi-definite matrix. Therefore,

$$(\mathbb{E}[M_t M_t^T])^2 \leq \|A\|_F^2 (\beta_1 - \sigma^2 \min(A)).$$

Similarly, we can obtain

$$(\mathbb{E}[M_t^T M_t])^2 \leq \|A\|_F^2 (\beta_2 - \sigma^2 \min(A)),$$

where,

$$\beta_2 = \max_j \sum_{i=1}^{m} \left\{ \frac{\alpha \|A\|_F^2}{|A_{ij}| \|A\|_1} + (1-\alpha) \right\}^{-1}, \quad \text{for } A_{ij} \neq 0.$$

We can now apply Theorem 1 with

$$\rho^2(\alpha) = \|A\|_F^2 \max\{\beta_1, \beta_2\} - \sigma^2 \min(A)$$

and

$$\gamma(\alpha) = \|A\|_1 \frac{\alpha + (1-\alpha)\lambda}{\|A\|_F^2} + \|A\|_2$$

to conclude that $\|S\Omega(A) - A\|_2 \leq \varepsilon$ holds subject to a failure probability at most

$$(m+n) \exp\left(-\frac{s\varepsilon^2}{2\rho^2(\alpha) + \frac{2}{3}\gamma(\alpha)\varepsilon} \right).$$

Finally, we can bound the failure probability by δ, and setting $\varepsilon = \epsilon * ||\mathbf{A}||_2$

7.2 IMPLEMENTATION

Our implementation of this algorithm is written in a jupyter notebook using numpy in python [3]. The data sets were downloaded and stored alongside the jupyter notebook so that it could be directly used by the python code for testing with the algorithm.

[3] https://github.com/ (will link to actual github repository in final version)

8 EXPERIMENTS

8.1 DATA SETS

While Kundu's original paper tested its sparsification and PCA approximation algorithms well, it didn't do so extensively. There are still many different types of data sets to test it with, so our paper has determined three more data sets that are worth testing on. Our choices of data sets each have their own justification for being selected.

We chose to further test the algorithm on a complete handwritten digit data set that includes zero through 9. While the original paper did test the algorithm on handwritten digits, it oddly chose to only include the numbers 1, 6, and 9 (Kundu, 2015). This could have been due to many different reasons such as a constraint on the amount of computing power available to this project. However, the inclusion of 6 and 9 specifically was likely not a coincidence. There is merit in seeing how the results of the algorithm on a complete handwritten digit set will compare to the paper's original results on a limited portion of the set.

As for the license plate image data set, we chose this data set for a few reasons. Firstly, it is simply an image data set which is perfect for being represented as matrices, and the results are able to be easily visualized to be better understood. Secondly, images of license plates form a sort of block matrix due to the license plate being its own section within the image. Block images are notoriously difficult for sparsification algorithms to deal with, so testing Kundu's algorithm against a real world data set of block matrices is important ().

We lastly chose the football match history data set because it is very different from the style of data tested against in the original paper. The results of matches in football can vary significantly, and the increments in which points are awarded creates an odd distribution of scores. It will be interesting to see how the algorithm is able to handle this different type of data and its range of values.

8.2 RESULTS

8.2.1 HANDWRITTEN DIGITS

We tested the sparsification algorithm on this data set by applying it to each element of the MNIST handwritten digit data set. We then compared the output of the sparsification to the original matrix by taking the norm of the difference between the two and dividing it by the norm of the original matrix. By using this method on each matrix, we can find the average accuracy of the algorithm on this data set. We will then do this repeatedly for different levels of sparsification (values of τ). We will also complete this same process for both the $l1$ and $l2$ sparsification algorithms so that we can compare the results of each.

At this point in our project, we have applied this process to the Handwritten digits data set with $\tau = 10$. The results are below, but they will be in the form of a graph where τ is the x axis and the y axis is the accuracy with each of the different results for the sparsification algorithms being plotted together.

We found that the average accuracy of the sparsification algorithm was 0.128 with $l1$ and $l2$ both having identical average accuracies of 0.369. This is a significant drop below the $l1$ and $l2$ algorithms, so it is a bit odd. These results are a bit better when the digits used are limited to the numbers 1, 6, and 9 which were 0.146 and 0.380 respectively. However, these results do not make too much sense, so it is likely that this may be due to the fact that our implementation of the algorithm needs to be debugged, but this is a problem that can be easily solved.

As for testing the fast PCA algorithm, we did so by applying it to each digit, and finding the accuracy by comparing it to the actual PCA result for each digit like we did for the sparsification algorithm. Once again, we will we create a graph like above where k is the x axis and accuracy is the y axis.

At this point in our project, we have applied this process to the Handwritten digits data set with $k = 4$. The results are below, but they will be in the form of a graph where k is the x axis and the y axis is the accuracy with each of the different results for the sparsification algorithms being plotted together.

We found that the average accuracy of the PCA algorithm to be off by 1.403 which is clearly not possible, so this is likely a problem the preliminary stages of our implementation of the algorithm and its tests. These results will make more sense once they are achieved correctly and compiled into a graph as described above.

8.2.2 LICENSE PLATES

We tested the sparsification algorithm on the License plate data set in much the same way that we did with the handwritten digit data set. The major difference with this data set is simply that the individual matrices are much larger due to the

higher resolution images. Also, these matrices are examples of block matrices that are generally difficult for sparsification algorithms to deal with.

We don't have these results yet, but it will simply be a matter of applying the algorithm in the same way we did above but with possibly different parameters after preprocessing the data.

We also tested the PCA algorithm on the license plate data set in much the same way that we did with the handwritten digit data set with once again the main major difference simply being the much larger size of the matrices in this data set.

We don't have these results yet, but it will simply be a matter of applying the algorithm in the same way we did above but with possibly different parameters after preprocessing the data.

8.2.3 Football Historical Scores

We also tested the sparsification algorithm on the football historical scores data set in much the same way that we did with the handwritten digit and license plate data sets. These matrices are between the handwritten digit and license plate matrices in terms of size.

We don't have these results yet, but it will simply be a matter of applying the algorithm in the same way we did above but with possibly different parameters after preprocessing the data.

We also tested the PCA algorithm on the Football Historical Scores data set in the same way as above.

We don't have these results yet, but it will simply be a matter of applying the algorithm in the same way we did above but with possibly different parameters after preprocessing the data.

9 Conclusion

The conclusion here depends on the results of the data, but, if we find the results to be promising with similar or better outcomes as in the original paper, then we will be supporting its conclusion and the algorithms it proposed. However, if we find unfavorable results, then this paper will conclude that the original algorithm is not as widely applicable as it may have originally been presented to be.

References

Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9–es, 2007.

Dimitris Achlioptas, Zohar Karnin, and Edo Liberty. Matrix entry-wise sampling: Simple is best. *Submitted to KDD*, 1: 1–4, 2013a.

Dimitris Achlioptas, Zohar Karnin, and Edo Liberty. Matrix entry-wise sampling: Simple is best. *Submitted to KDD*, 1: 1–4, 2013b.

Sanjeev Arora, Elad Hazan, and Satyen Kale. A fast random sampling algorithm for sparsifying matrices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006. Proceedings*, pp. 272–279. Springer, 2006.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Petros Drineas and Anastasios Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 111(8):385–389, 2011a.

Petros Drineas and Anastasios Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 111(8):385–389, 2011b.

Abhisek Kundu. *Element-wise matrix sparsification and reconstruction*. Rensselaer Polytechnic Institute, 2015.

Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

A APPENDIX

Github repository with code used in this paper: https://github.com/