RESEARCH ARTICLE | FEBRUARY 16 2024

Binary data in matrices with singular value decomposition method

Rihab H. Sahib

AIP Conf. Proc. 3051, 040011 (2024)
https://doi.org/10.1063/5.0191716

Binary Data in Matrices with Singular Value Decomposition Method

Rihab H. Sahib[a)]

College of Arts, University of Babylon, Babylon, Iraq

[a)]Corresponding Author: art.rehab.habeeb@uobabylon.edu.iq

Abstract. A big challenge that faces many applications in different fields suffers in dealing with datasets of massive size. Additionally, retrieving and casting this data is somewhat time-consuming. Applications such as government or any institution election, surveys, healthcare …etc., leverage techniques of data reduction, dimensionality reduction, matrix decomposition, or compression such as the Singular Value Decomposition Technique. Our paper shows the use of this technique as a method in certain circumstances where data is of binary type and can be retrieved, cast, or updated in less time and in a smaller size without losing any information. In other words, we prove practically that the massive size of binary values can be managed in a form of matrices with low rank (low rank is one of the bases used in the Singular Value Decomposition technique) to return the exact matrix of information instead of dealing with the original large matrix of data. The experimental results are implemented on a Lenovo machine, Intel Corei5, CPU 2.5GH with 8GB of RAM, using visual basic, C#, in Visual Studio 2019 environment.

INTRODUCTION

One of the data mining methods which are used to decompose and analyze a high-dimension matrix of data is the Singular Value Decomposition technique (SVD) which results in a low-dimensional representation of that matrix of data. Eliminating parts of data that are less or not important which makes it easier to represent with any desired number of dimensions [1].

SVD is the basis of machine learning and data mining in many fields [2]. It makes it simple to ignore data that are not necessarily needed (which is zero in this research, as it deals with binary data of zeros and ones) [3].

The general base of this technique is to divide a given matrix of input data into three vectors (U, S, and the transpose V), then apply the product operation to these vectors such that $X = USV^T$ (where T is the transpose of vector V). U and V^T are unitary matrices that produce a rotation of the input data. S is a diagonal matrix of singular values [1, 4].

Any matrix of a two-dimension form A of size $m \times n$ can be factorized into three matrices, m refers to rows, n refers to columns, and "$m \geq n$" [5, 6]. Multiplying the three matrices produce an approximately equal matrix to the original matrix A, as explained in equations (1), (2), and (3) [6, 7].

$$A = USV^T \tag{1}$$

$$A = \begin{bmatrix} u_{1,1} & \cdots & u_{1,n} \\ u_{2,1} & \cdots & u_{2,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & & \vdots \\ u_{n,1} & & u_{n,n} \end{bmatrix} \begin{bmatrix} \sigma_{1,1} & 0 & \cdots & 0 \\ 0 & \sigma_{2,2} & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \vdots & & \cdots & \vdots \\ \vdots & & & \vdots \\ 0 & & & \sigma_{n,n} \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{1,n} \\ v_{2,1} & \cdots & v_{2,n} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & & \vdots \\ v_{n,1} & & v_{n,n} \end{bmatrix} \tag{2}$$

$$A = \sum_{i=1}^{n} \sigma_i \, u_i \, v_i^T \tag{3}$$

3rd International Conference on Engineering and Science
AIP Conf. Proc. 3051, 040011-1–040011-12; https://doi.org/10.1063/5.0191716
Published by AIP Publishing. 978-0-7354-4808-7/$30.00

U and V^T are vector matrices where: $U^TU= UU^T= I$ (I is the identity matrix), and $V^TV=VV^T=I$, and S is a diagonal matrix where only the diagonal singular values are non-zero values in which: $\sigma 1 \geq \sigma 2 \geq \sigma 3 .... \geq \sigma m \geq 0$.

When $n \geq m$, the matrix S has at most m non-zero elements on the diagonal. Therefore, it is possible to exactly represent the original matrix A using the economy SVD as shown in equation (4) which is denoted as shown in equation (5) (Economy SVD) [8].

$$\sigma_1 u_1 v_1 + \sigma_2 u_2 v_2 + ... + \sigma_m u_m v_m \tag{4}$$

$$A = \hat{U}\hat{S}V^T \tag{5}$$

In equation (4) $r=1$, and if there are very low singular values σ (the number of non-zero singular values represents the rank (r)), then the low singular values σ (zero or less) can be truncated and the truncated SVD may still be exact [9] as written in equation (6) (Truncated SVD) Where:

$$A = \overset{\approx}{U}\overset{\approx}{S}V^T \tag{6}$$

With rank $k \leq m$ (k is an estimated value chosen several times practically in this study to find the best rank that retrieves the exact matrix after applying the SVD technique which separates' the original matrix into three different forms that can be, for example, stored in three different places such as Clouds …etc.), the left singular vector (U) is $n*k$, the right singular vector (V^T) is $k*m$, and the matrix of singular values is the sub-block of $k*k$.

In the truncated SVD the property of $\tilde{U}^T \tilde{U} = Ik*k$, but $\tilde{U}\tilde{U}^T \neq Ik*k$ because the "identity matrix" is of size n*n which is not true for $Ik*k$ [8]. Applying SVD on a huge matrix (resulting in U, S, and V^T) can be retrieved in less time by multiplying the three small matrices (after truncating unimportant values), based on the best low-rank approximation, as shown in Fig. 1.

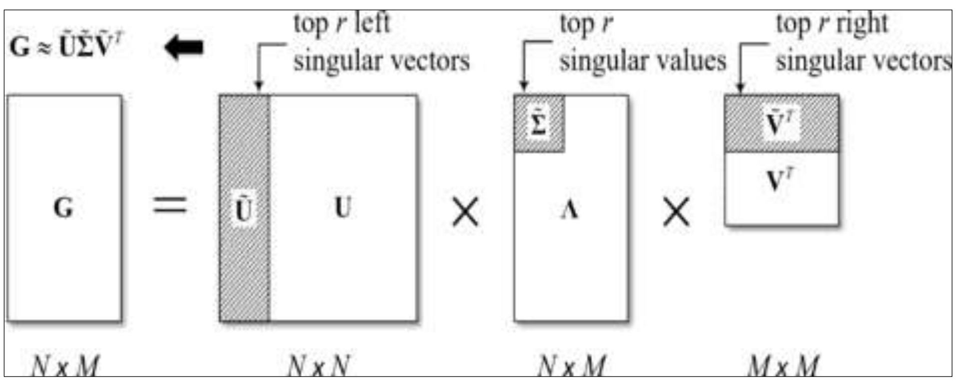

FIGURE 1. Size reduction of applying SVD on matrix G resulting in three small separated matrices

In the figure above the shadow, area means the low-rank singular value matrix and singular matrices, moreover, the area size can reflect the number of matrix elements [10]. In many applications, SVD provides a robust tool when the objective is the exact Binary Matrix Decomposition (BMD) [11]. For $A \in \{0, 1\}\ m\times n$, we note that A is an $m \times n$ Binary matrix. BMD aims to find two matrices "$U \in \{0, 1\}m\times k$ and $V \in \{0, 1\}k\times n$ such that the difference $\{M - U \circ V\}L$ under some norm L is minimized with a given k as small as possible". The minimum possible k is called the Boolean rank of a binary matrix A which may be smaller or larger than its real rank [12].

The exact BMD is satisfying and pleasing and it is useful for many applications in the future. However, it can be used for approximate BMD, by finding the product of $U \circ V$ that covers most of the ones and no zeros in A. This is sometimes called the “from-below” approximation [13].

An Algorithm that illustrates the general SVD and how this data mining technique generates three different matrices reducing the size of the original matrix without losing important information is shown in Fig. 2.

Algorithm 1rSVD

Require: $\boldsymbol{A}$ (real $m \times n$ matrix, without loss of generality assume $m \leq n$), k (desired rank of truncated SVD), p (parameter for oversampling dimension), $\ell = k + p < m$ (dimension of the approximate column space), q (exponent of the power method)

Ensure: Approximate rank-k SVD of $\boldsymbol{A} \approx \widehat{\boldsymbol{U}}_k \widehat{\Sigma}_k \widehat{\boldsymbol{V}}_k^{\mathrm{T}}$

Generate an $n \times \ell$ random matrix $\boldsymbol{\Omega}$.

Assign $\boldsymbol{Y} \leftarrow (\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^q \boldsymbol{A}\boldsymbol{\Omega}$.

Compute $\boldsymbol{Q}$ whose columns are an orthonormal basis of $\boldsymbol{Y}$.

Compute the SVD of $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A} = \widehat{\boldsymbol{W}}_\ell \widehat{\Sigma}_\ell \widehat{\boldsymbol{V}}_\ell^{\mathrm{T}}$.

Assign $\widehat{\boldsymbol{U}}_\ell \leftarrow \boldsymbol{Q}\widehat{\boldsymbol{W}}_\ell$.

Extract the leading k singular vectors and singular values from $\widehat{\boldsymbol{U}}_\ell$, $\widehat{\Sigma}_\ell$ and $\widehat{\boldsymbol{V}}_\ell$ to obtain $\widehat{\boldsymbol{U}}_k$, $\widehat{\Sigma}_k$ and $\widehat{\boldsymbol{V}}_k$.

FIGURE 2. SVD algorithm resulting the three matrices that reduce the size of the original matrix

SVD APPLICATIONS

The following sections illustrate the SVD data mining technique dealing with data in matrices of different values as a data reduction and dimensionality reduction tool.

Low-Rank Approximation of Matrices

Many fields of science use SVD to get the effective low rank that can decompose data separately such as data compression, image processing, engineering, and approximating a matrix by a rank that is low as possible matrix according to a norm that is given previously[15, 16].

Image Compression Using the Singular Value Decomposition

In general, the SVD decomposes the original matrix into three matrices. The aim is to approximate the data set of high dimensions using fewer dimensions. SVD displays the substructure of the high-dimensionality original data by reducing it into a lower-dimensional matrix and arranging the data from the most to the least variation[17].

SVD factorizes the $m*n$ original matrix into three matrices, written as $A = U\Sigma V^T$ where U_{m*m} and V^T_{n*n} are orthogonal matrices known as left and right singular vectors of A respectively and Σ is a diagonal matrix of real numbers that are non-negative numbers known as singular values of A in the order $m\times n$ [18].

The SVD of a given matrix can be calculated as follows:

- From a given matrix A, calculate AA^T and A^TA (A^T is the transpose matrix of A).
- Use AA^T to form U, which is calculated by calculating eigenvalues and eigenvectors of AA^T.
- V can be formed by calculating the eigenvalues and eigenvectors of A^TA, in the same manner.
- U and V^T columns are produced by dividing each eigenvector by its magnitude.
- The square root of eigenvalues results in the singular values which are arranged in descending order in the diagonal matrix.

Determination of the Effective Rank

SVD can be employed to detect the actual and the numerical rank of a matrix, by counting the number of singular values that are above a certain tolerance (t). The tolerance τ =0 is used for the actual rank and some small number determined by the user according to the application at hand for the numerical rank (i.e., τ>0 for numerical rank) (e.g., τ = ε ||A||2= ε 1 where ε is machine precision). The numerical rank of a matrix is defined as the number of singular values ε > τ, r(A) τ = { k: k (A) > τ, k+1 (A) ≤ τ } [5].

EMPLOYING SVD in BINARY MATRICES

Employing Singular Value Decomposition to Binary matrices can work efficiently, as data are exactly either 0 or 1. Which eliminates dealing with a matrix of massive size. Binary data can be managed using SVD with rank $k<r$ in which r is the actual rank for a given matrix.

In special cases where the matrix of data should contain at most only one value of 1 in a column (under special circumstances like election events) such that each time the matrix is retrieved, the SVD is applied on a few columns incrementally, the rank can be as low as possible.

We illustrate SVD in a special case on data for an e-voting system, and generally for matrices of binary form that can be used in all types of surveys, voting systems, symptoms of disease…etc.

Special Case of SVD with Matrices of Binary Data

An e-voting system that merges the concept of a distributed ledger with the Singular value decomposition can treat the matrix (ledger) of zeros and ones as an incremental ledger in which a structure of blocks is used to contain transaction of votes where m (the columns) represents the number of candidates, and n (the voters) represents the number of voters [19].

SVD is applied on blocks of transactions (votes) where each block contain a fixed number of transactions. Under this circumstance, SVD manages data for each block to result in a ledger incrementally. In other words, SVD is initially applied on a matrix of zero values every time a block of few transactions arrives. Compared with the overall size of the matrix, SVD needs a low rank to work on each block incrementally.

For example, if a matrix A is of size $m*n$, where $m<n$, under the following conditions:

- Each column (refers to the voter) should have at most only one value of 1 (referring to a vote corresponding to a candidate, where rows are candidates).
- The data for several columns are treated as a block of transactions (votes), in which a few columns in the matrix are changed to achieve condition (1).

SVD in such case work on fewer data incrementally, then SVD can be applied efficiently by $r= k$, where r is the actual rank in which $r<m$ and k is the lowest rank based on the number of transactions in a block where $k\leq r$.

The size of the matrix that is retrieved after applying SVD is $m*k + k*k + n*k$ which is lower than the $m*n$. We say a special case because SVD is applied gradually on a few columns instead of all columns, as these few columns represent voters who voted for candidates to perform a transaction in a block. In such a case the rank can be lower than the actual rank of the overall matrix which is a good choice for SVD of binary matrices to be employed with blockchain technology or any other distributed ledger technology.

For example, A ledger of 8*10 as shown in Table 1 is a simple example where of values for ones and zeros, each column has only one value of 1, which refers to a vote for the desired candidate. The following ledger, is a copy of the SQL database for results, (V) refers to the voter, (C) refers to the candidate, voter1 (V1) voted for candidate8 (C8), V2 voted for C2, and so on.

SVD works on this ledger to transform the values into another form and distribute the outputs in three storage nodes that are cast to different servers in the world.

TABLE 1. Original Matrix (ledger) of size 8*10.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
C1	0	0	1	0	0	0	0	0	0	1
C2	0	1	0	0	0	0	0	1	0	0
C3	0	0	0	0	0	1	0	0	0	0
C4	0	0	0	0	0	0	0	0	0	0
C5	0	0	0	1	0	0	0	0	0	0
C6	0	0	0	0	0	0	0	0	0	0
C7	0	0	0	0	1	0	0	0	0	0
C8	1	0	0	0	0	0	1	0	1	0

First, the ledger is checked to insure validation by having one value of the form (1) in each column which is a vote for the preferred candidate. The outputs of SVD are three matrices (*U*, *S*, and V*T* matrices). The result of applying SVD for this example is shown in Table 2, Table 3, and Table 4.

TABLE 2. The left matrix U results from applying SVD on an 8*10 matrix (ledger).

0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	-1
0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	-1	0	0	0
-1	0	0	0	0	0	0	0

TABLE 3. The diagonal singular value matrix S results from applying SVD on an 8*10 matrix (ledger).

1.732	0	0	0	0	0	0	0	0	0
0	1.4142	0	0	0	0	0	0	0	0
0	0	1.4124	0	0	0	0	0	0	0
0	0	0	1.0000	0	0	0	0	0	0
0	0	0	0	1.0000	0	0	0	0	0
0	0	0	0	0	1.0000	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

TABLE 4. The right matrix V S results from applying SVD on an 8*10 matrix (ledger).

-0.5774	0	0	0	0	0	0	0.5164	-0.1225	-0.6205
0	0.7071	0	0	0	0	-0.7071	0	0	0
0	0	0.7071	0	0	0	0	-0.5477	-0.0866	-0.4387
0	0	0	1.0000	0	0	0	0	0	0
0	0	0	0	-1.0000	0	0	0	0	0
0	0	0	0	0	1.0000	0	0	0	0
-0.5774	0	0	0	0	0	0	-0.2582	-0.6325	0.4472
0	0.7071	0	0	0	0	0.7071	0	0	0
-0.5774	0	0	0	0	0	0	-0.2582	0.7550	0.1733
0.0000	0	0.7071	0	0	0	0	0.5477	0.0866	0.4387

These matrices *U*, *S*, and V^T (V transpose) transformed the general ledger into coefficients of another form, resulting in an adaptive SVD ledger that can be distributed and stored in three separate countries using cloud storage services and servers.

Each matrix is separated from the other and can be dealt with as a decomposed matrix, which the main aim of SVD is to reduce the dimensionality of the original matrix without losing information.

This is done by choosing a rank for the matrix depending on the singular values in matrix *S* (the diagonal matrix), in which we can use a low rank to deal with the three matrices more efficiently. Since our data is of binary type, which is suitable for elections, the rank can be as low as possible as long as it returns the exact matrix.

The rank of *r* or fewer results in an acceptable decomposition for large data sets, the following equation (7) is used to show how we deal with the size of large data sets of the binary form :

$$U(:,1:r)*S(1:r,1:r)*V(:,1:r)^T \tag{7}$$

Where U(:, 1:r) means keeping all the rows denoted by (:) and only from 1 to *r* denoted by (1:r) columns of *U*, S(1:r, 1:r) refers to all the rows and columns from 1 to r (1:r) of *S*, and $V(:,1:r)^T$ referring to all the rows (:) and only columns from 1 to r (1: r) of V^T.

This calculation leads to the fact that we can deal with decomposed matrices of less size instead of a ledger size 80 (8*10). For example, if the rank of 3 is used, then we can deal with a matrix of size 63 (8*3 + 3*3 + 10*3) resulting from equation (8):

$$U*r + r*r + V^T*r \tag{8}$$

Which makes a massive difference compared with the original matrix. We can retrieve the original matrix by applying $U*S*V^T$ resulting in the same ledger with no loss of information as shown in Table 5 which is similar to the original general ledger shown in Table 6.

TABLE 5. Retrieving the original matrix.

0	0	1.0000	0	0	0	0	0	0	1.0000
0	1.0000	0	0	0	0	0	1.0000	0	0
0	0	0	0	0	1.0000	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1.0000	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1.0000	0	0	0	0	0
1.0000	0		0	0	0	1.0000	0	1.0000	-0.0000

TABLE 6. The original matrix.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
C1	0	0	1	0	0	0	0	0	0	1
C2	0	1	0	0	0	0	0	1	0	0
C3	0	0	0	0	0	1	0	0	0	0
C4	0	0	0	0	0	0	0	0	0	0
C5	0	0	0	1	0	0	0	0	0	0
C6	0	0	0	0	0	0	0	0	0	0
C7	0	0	0	0	1	0	0	0	0	0
C8	1	0	0	0	0	0	1	0	1	0

General Case of SVD with Matrices of Binary Data

In a scenario where SVD is applied on all m and n at once for a matrix A, r is the actual rank, as shown in equation 9.

$$r = \begin{cases} r < m & ,if\ m < n \\ r \leq n & ,if\ n < m \end{cases} \tag{9}$$

Whether r is the actual rank or less, in both ways, it is acceptable to deal with the decomposed matrices instead of dealing with a binary matrix of massive size.

The experimental results are illustrated in Table 7 at the end of this section, comparing matrices of binary data once when $m<n$, another when $n<m$, and when $m=n$. Also, exhibiting the time needed for generating a matrix under the condition of having one value of 1 randomly in each column all at once, the time needed for applying SVD, the suitable rank, and the time needed for extracting back the matrix by multiplying $U*S*V^T$.

EXPERIMENTAL RESULTS

The experimental results show several tests of SVD with Binary matrices once when $m<n$, other tests when $m>n$, and when $m=n$. The results show the following:

- The time needed to generate the matrix using the randomization function that sets only one value of 1 randomly in each column before applying SVD.
- The time needed for SVD to be applied on the original matrix.
- The suitable rank r for retrieving the exact matrix.
- The time needed after applying SVD and extracting the original matrix back. The significance lies in the time it takes for SVD to be applied, constructed, and extract the matrix to match the original matrix. Table 7 shows the results respectively according to the size of the binary matrix[see the end of this section].

The Algorithm below shown in Fig.3 illustrates the SVD technique with Binary data of a matrix in three cases based on the entered number of columns and rows, a case when the number of rows (m) is less than the number of columns (n), a case when the number of rows (m) is larger than the number of columns (n), and another case we enter the same number for column and rows m=n.

The program shows us the input matrix after we entered the number of columns and rows, then the SVD algorithm is applied to result in three decomposed matrices, U which is m*r (r refers to the rank which is the number of non-zero singular values in the decomposed matrix S), S is an r*r matrix and V which is r*n transposed matrix.

After multiplying the three decomposed matrices ($U*S*V^T$) we find that even when the size is reduced and separated into smaller dimensions, the result is the same original matrix. That leads to one of the useful benefits of SVD as it aims to save important data in another form in three decomposed matrices.

Algorithm 2 SVD with binary data

Require: An original matrix m*n, number of columns m, number of rows n

Ensure: Time needed to generate the original binary matrix, the time needed for SVD to decompose the matrix, the rank, the time needed to apply SVD ($U*S*V^T$)

1. Begin
2. Generate a binary matrix using a random function (depending on the entered number of columns m and rows n)
3. Apply SVD decomposition on the matrix
4. resulting in three new small matrices (U, S, V^T) and the value of rank (r)
    a. For m=1 to r U is generated
    b. For s= 1 to r S is generated
    c. For n=1 to r V^T is generated (Transposed automatically when using the SVD algorithm)
5. Multiply $U*S*V^T$ to extract the new matrix
6. The original matrix equals the output of ($U*S*V^T$)
7. Message showing the time of each step
8. End

FIGURE 3. SVD applied on a matrix of binary data

Experimental Results of SVD for Binary Matrices when m<n

An input matrix of binary data in which the number of columns m=10, and rows n=100 is shown in Fig. 4, which is a simple program written in a visual studio 2019 environment. It shows the binary matrix on the left, a notification bar below showing the time it takes for each step, and the extracted matrix after applying SVD to retrieve the original matrix on the right. It is seen that the time needed to generate the matrix using the randomization function that sets only one value of 1 randomly in each column before applying SVD is 1.62 ns, the time needed for SVD to be applied on the original matrix is 222.3 ns, the suitable rank r for retrieving the exact matrix is 8, and the time needed after applying SVD and extracting the original matrix back is 11.7 ns.

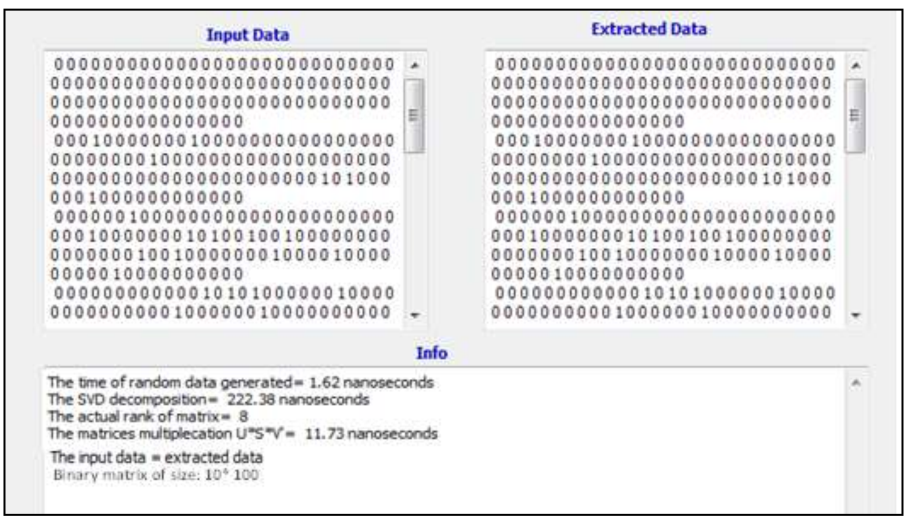

FIGURE 4. Binary matrix of size 10*100 with rank 8.

The same procedure is repeated on binary matrices in which $m<n$ and tested randomly each time in different ranks, details are shown in Table 7 at the end of the experimental results.

Experimental Results of SVD for Binary Matrices when M>N.

An input matrix of binary data in which m=100, and n=10 is shown in Fig. 5, showing the needed results. After repeating the tests randomly on the same matrix, in which the values of 1s are spread randomly in the binary matrix, it is seen that the different number of ranks for each test is always $r \leq n$ and can return the exact original matrix based on how the 1s are spread randomly due to the randomization function that is used for this task with a rank of 10.

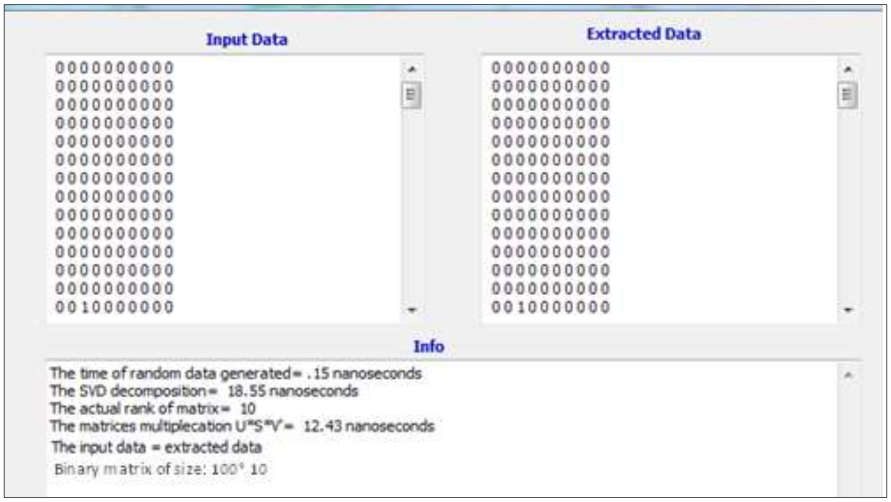

FIGURE 5. Binary matrix of size 100*10 with rank 10.

The same procedure is repeated on binary matrices in which $m>n$ and tested randomly each time in different ranks, details are shown in Table 7 at the end of the experimental results.

Experimental Results of SVD for Binary Matrices when M=N.

In an input matrix of binary data when m and n =100, it is shown in Fig. 6 that the time needed to generate the matrix using the randomization function to replace one zero by 1 randomly in each column is 0.98ns. The time needed for SVD to be applied on the original matrix is 937.57ns. A suitable first rank r for retrieving the exact matrix is 67. Other ranks were 62, 64, 65, 60, and 63 are shown in Table 7 at the end of the experimental results. The time needed after applying SVD and extracting the matrix is 746.34ns.

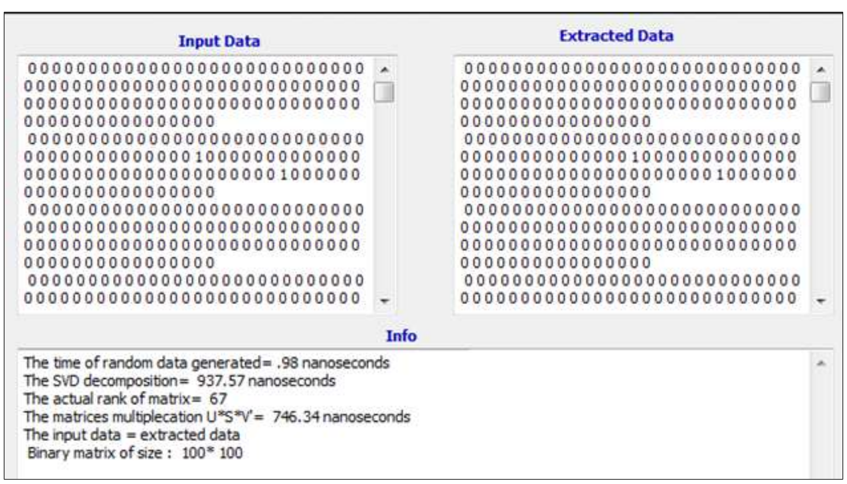

FIGURE 6. Binary matrix of size 100*100 with rank 67.

Table 7 exhibits the experimental results that retrieve the exact original matrix, some matrices were rested several times, and each time the rank differs according to the values of 1 that are randomly located in each column using the random equation.

TABLE 7. Exhibiting the experimental results.

	Binary matrix (M<N)	**The time needed to generate the matrix**	**The time needed for SVD**	**suitable ranks**	**The time needed to retrieve original matrix $U*S*V^T$**
1	10*100	1.62ns	222.38ns*	8	11.73ns
2	50*100				
	Test1	0.99ns	203.28ns	39	224.33ns
	Test2	0.6ns	196.38ns	44	136.85ns
	Test3	0.76ns	352.3ns	42	232.05ns
	Test4	0.59ns	303.17ns	43	97.74ns
	Test5	0.45ns	140.52ns	46	299.8ns
	Test6	0.63ns	201.45ns	40	205.62ns
3	5*1000	3.54ns	64.91ns	3	22.03ns
4	100*1000	8.86ns	6.464µs*	98	4.736 µs
5	100*100000	758.59ns	1.9174ms*	98	0.656ms
6	100*10				
	Test1	1.17ns	18.57ns	7	10.31ns
	Test2	0.15ns	18.55ns	10	12.43ns
	Test3	0.11ns	20.75ns	9	11.34ns
	Test4	0.15ns	18.83ns	8	10.35ns
7	100*50				
	Test1	1.51ns	712.97ns	39	216.28ns
	Test2	0.49ns	299.53ns	36	212.69ns
	Test3	0.51ns	309.82ns	40	232.4ns
	Test4	0.47ns	315.33ns	37	190.27ns
	Test5	0.49ns	271.78ns	34	112.74ns
	Test6	0.48ns	231.24ns	38	211.83ns
8	1000*5	0.2ns	60.83ns	5	31.14ns
9	1000*10(				
	Test1	3.9ns	5.5146µs	96	4.1326 µs
	Test2	2.63ns	5.1909µs	92	4.2357 µs
	Test3	2.9ns	5.3805µs	95	4.4833 µs
	Test4	1.85ns	5.5251µs	93	4.1490 µs
	Test5	2.96ns	5.5490µs	97	4.3561 µs
	Test6	2.78ns	5.4024µs	91	4.10464 µs
10	100*100				
	Test1	0.98ns	937.57ns	67	746.34ns
	Test2	0.93ns	951.27ns	62	514.8ns
	Test3	0.95ns	579.82ns	64	631.9ns
	Test4	0.85ns	613.66ns	65	475.99ns
	Test5	0.92ns	835.36ns	60	633.98ns
	Test6	0.91ns	652.66ns	63	711.19ns
11	500*500				
	Test1	11.5ns	55.565µs	312	38.174 µs
	Test2	15.83ns	55.241µs	307	37.788 µs
	Test3	21.72ns	55.011µs	328	41.464 µs
	Test4	12.9ns	5.543µs	314	39.766 µs
	Test5	11.79ns	57.305µs	329	42.773 µs

CONCLUSION

In this paper, we illustrated different binary matrices with values {0,1} of large size that can be dealt with within less dimensionality based on a low rank that returns the exact matrix. Singular value decomposition with rectangular binary matrices where $m<.n$ can have rank $r<m$ in which r is less than m to extract the exact or approximate matrix.

In special cases where SVD is employed within distributed ledger in which the ledger of rectangular binary data is constructed incrementally, the rank can be less than m because it deals progressively with a low amount of data compared with the whole massive size of A incrementally depending on the number of transactions in each block. When $m>n$ the rank is $r\leq n$ in which r can be at most equal or no more than n to extract the exact or approximate matrix.

In Binary matrices, whether the matrix is of rectangular or square data, the rank is automatically chosen based on the smallest number of rows n or columns m.

According to the experimental results, applying SVD on binary matrices with massive sizes does not exceed even one second, which makes it useful in applications for such types of data.

Whenever the matrix is large and of binary form it can be decomposed according to the rank that can be less or equal to n or m to retrieve or cast matrices in a decomposed manner instead of matrices with size $n*m$. The experimental results show that whenever n or m is much smaller than each other, the chances of having several ranks to extract the exact matrix are very low. Also, it is shown that SVD with rectangular binary matrices takes less time than square binary matrices to be applied and extracted.

FUTURE WORK

Employing SVD as a data and dimensionality reduction tool is suitable for massive data that are formed as matrices, in which data can be retrieved, stored, or cast within less time and space. SVD is an acceptable choice for all types of data as matrices. Especially if dealing with large data of binary form in healthcare applications indicating symptoms of disease, Education systems, Government applications such as e-voting systems, surveys, etc

REFERENCES

1. B. Justyna, "Singular Value Decomposition Approaches in A Correspondence Analysis with the Use of R", Folia Oeconomica Stetinensia, 18,178-189 (2018).
2. K. Hiroaki, "RGB to spectral image conversion using spectral pallete and compression by SVD", Proceedings 2003 Int. Conference on Image Processing, 2,461–464, (Barcelona, Spain, 2003).
3. L. Jure, R. Anand, and U. Jeff, "Mining of massive datasets", (Cambridge University Press, UK, 2014).
4. F. Mario and M. Joachim," Selecting the rank of truncated SVD by maximum approximation capacity", IEEE Int. Symposium on Information Theory Proceedings,1063-1040, (St. Petersburg, Russia, 31 July-5 Aug 2011).
5. K. Ashish, K. Anil, and K. Prabin, "Enhancement of Low Contrast Satellite Images using Discrete Cosine Transform and Singular Value Decomposition", World Academy of Science, Engineering, and Technology J.,5, 707- 713 (2011).
6. K. Meenakshi, R. Srinivasa, P. Satya, "A Fast and Robust Hybrid Watermarking Scheme Based on Schur and SVD Transform", Int. J. of Research in Engineering and Technology, 3, (2014).
7. B. Madhu and H. Ganga, "An optimal and secure watermarking system using SWT-SVD and PSO", Indonesian J. of Electrical Engineering and Computer Science, 18, 917-926 (2020).
8. L. Steven and K. Nathan, "Singular Value Decomposition" in *Data-driven science and engineering*, (University of Washington, US, 2017).
9. S. Yuan, Y. Shiwei, S. Yi and K. Tsunehiko, "Exact and approximate Boolean matrix decomposition with the column-use condition", Int. J. of Data Science Analysis, 1, 199–214 (2016).
10. Tingzhao Wu, Ruimin Hu, Xiaochen Wang and Shanfa Ke, "Multimedia Tools and Applications ", Springer Science+Business Media, LLC, part of Springer Nature 2019, 78, 20723–20738 (2019).
11. D. Gregory and N. Pullman, "Semiring rank: Boolean rank and nonnegative rank factorizations", J. Combinatorics Information and System Sciences, 8,223–233 (1983).

12. R. Bˇelohlávek and M. Trneˇcka, "From-below approximations in Boolean matrix factorization: geometry and a new algorithm", J. of Computer and System Science, 81, 1678–1697 (2015).
13. N. Shireen, "Singular Value Decomposition of Rectangular Matrices", MSc thesis, An-Najah National University, Nablus, Palestine, 2009.
14. T.-L. Chen, S.-Y. Huang and W. Wang, "A consistency theorem for randomized singular value decomposition", Statistics and Probability Letters 161 (2020).
15. H. Swathi, S. Shah, Surbhi and Gopich, G., " Image compression using singular value Decomposition", IOP Conference Series: Materials Science and Engineering, 263, (2017).
16. K. Neethu and J. Sherin, "Using Approximate K-SVD Algorithm", Embedded and Communication Systems, Int. Conference on Innovations in Information, (Coimbatore, India, 2015).
17. A. Abdalkareem and T. Alasady, "Geo-Localization of Video-Based on Proposed LBP-SVD Method", Int. J. of Civil Eng. and Tech., 10, 407-423 (2019).
18. A. Abdalkareem and T. Alasady, "Proposed a Content-Based Image Retrieval System Based on the Shape and Texture Features", Int. J. of Innovative Technology and Exploring Engineering, 8, 2185- 2192 (2019).
19. R. H. Sahib and E. S. Al-Shimary, "An Online E-voting System based on an Adaptive Ledger with Singular Value Decomposition Technique", Karbala Int. J. of Modern Science, 7, 281-300 (2021).
20. R. Habeeb and E. Salih, "Developing an Adaptive SVD-based Distributed Ledger of an E-voting System", MSc thesis, Babylon University, 2021.