DOI https://doi.org/10.15407/csc.2023.03.015
UDC 681.3.062

O.O. MARCHENKO, Doctor (physical and math.), Professor, Head of the department, International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine, ORCID: https://orcid.org/ 0000-0002-5408-5279,
Glushkov ave., 40, Kyiv, 03187, Ukraine,
omarchenko@univ.kiev.ua

E.M. NASIROV, PhD (physical and math.), Senior Research Associate, International Research and Training Center for Information Technologies and Systems of the NAS and MES of Ukraine,
ORCID: https://orcid.org/0009-0006-9016-2602,
Glushkov ave., 40, Kyiv, 03187, Ukraine,
enasirov@gmail.com

METHODS OF DIMENSIONS REDUCTION IN TEXT PROCESSING ALGORITHMS

Paper describes methods of dimensionality reduction widely used in artificial intelligence in general, and in computer linguistics in particular, such as Non-negative matrix factorization and Singular value decomposition from the point of use in methods of Latent Semantic Analysis and Method of Principal Components. Advantages and disadvantages of each method are given. The computational complexity was investigated and a comparison of performance on dense and sparse matrices of different sizes was made. It is proposed to use them to reduce the dimensionality also of multidimensional linguistic data arrays.

Keywords*: artificial intelligence, computational linguistics, parallel computations.*

Introduction

Dimensionality reduction methods are a very important step in the text clustering process. Solving such a problem is difficult due to the high dimensionality of the data. Such a space with large dimensions reduces the efficiency of the process in general. The idea of these methods is to reduce the size of existing objects, turning them into a new space with low-dimensional objects.

Modern and widely used methods of dimensionality reduction are such methods as NMF and SVD [1–5]. This paper examines these methods to show their advantages and disadvantages from the point of usage in latent semantic analysis algorithms and the method of principal components. In text processing and in the theory of pattern recognition, cluster analysis consists in dividing a data set into several topics. Each topic, also called a cluster, contains data sets that are similar to each other and different from other groups.

Similarly, when clustering texts, they are divided into groups or topics. Last years, many researchers have conducted research in this field because of its importance in the field of data extraction and searching in a huge amount of text documents. In the process of clustering, the information is provided without pre-processing, so the task is to combine the given data set into important topics. Thus, clustering divides a given set of texts into M topics (clusters) so that the texts of one topic are "similar" and "different" from the texts of other topics. There are many real-world applications for the clustering process, including search engines and recommendations systems.

Latent Semantic Analysis

Latent semantic analysis (LSA), also known as latent semantic indexing (LSI) is the main method used to analyze the relations between documents and terms (words, n-grams) in the collection and to extract high-level concepts and transform the presentation of documents according to the identified relations. This is achieved by reducing the factors of space terms-documents. Terms can be both words and their combinations, the so-called n-grams, documents – ideally: sets of thematically homogeneous texts, or simply any, preferably, voluminous text (several millions of word forms), arbitrarily divided into pieces, for example paragraphs.

The LSA concept was patented in 1988 by a group of researchers at Bell Communications Research [6, 7]. The idea behind LSA was to overcome methods that only try to match search queries to the words from the document. Although this may seem sufficient for the purpose of searching for relevant documents, an intuitive approach of searching should be based on the conceptual content of the documents. LSA tries to overcome this problem by statistical analysis of hidden document structures. Thus, building a search engine that discovers meaningful relations is a common goal to solve the problem of incompetent search results. However, a general limitation of LSA is that there are cases where words have multiple meanings or polysemy.

LSA maps documents and individual words into the so-called "semantic space", in which all further comparisons are made. To construct the semantic space, a term-document matrix is used, which contains the frequency of occurrence of terms in documents. To construct a semantic space for a language, LSA firstly transforms a large representative text corpus into a rectangular term-document matrix, each cell of which contains the number of times a given word appears in a given document.

At the same time, the following assumptions are made:

1. A document is just a set of words. Words order in documents is ignored.

2. The only thing that matters is how many times the word appears in the document.

3. The semantic value of the document is determined by a set of words that usually are used together.

4. Each word has a single meaning. This is, of course, a strong simplification, but it is what makes the problem solvable.

The term-document matrix X, which describes the frequency of terms, is used as the source information.

Let's suppose we have some educational sample of texts. Let's present it in the form of a matrix A, the columns of which are a_i – vectors of terms, n – the number of terms. The term vector t_j is a column vector:

$$\vec{a}_j = \{a_{1j}, a_{2j}, \ldots, a_{mj}\},$$

where m is the number of documents in the training sample. The frequency of occurrence of the term in the document is equal to the number of occurrences of the term t_j in the document d_i:

$$a_{ij} = tf(t_j, d_i).$$

Then the resulting matrix is decomposed in such a way that each document is represented as a vector, the value of which is the sum of the vectors corresponding to its component words. Similarities between words and words, documents and words, and documents with documents are calculated as scalar products, cosines, or other vector-algebraic metrics.

LSA commonly uses a long-known matrix algebra method, singular matrix decomposition (SVD), which became widely used only after the advent of powerful digital computers and algorithms for their use in the late 1980s.

However, recently, usage of another approach, namely the use of non-negative matrix factorization (NMF Non-negative Matrix Factorization), is gaining more and more popularity due to the possibility of additional interpretation of the obtained non-negative results.

Principal Component Analysis

The method of principal components analysis (PCA) is one of the main ways to reduce the dimensionality of data with the loss of the minimum amount of information, developed by Karl Pearson in 1901. It is used in many areas, such as pattern recognition, computer vision, data compression, etc.

The main task of the method of principal components is to replace the original data with some aggregated values in a new space, while solving two tasks – the first of which is to combine the most important (from the point of minimizing the mean square metric) values into a smaller number of parameters, but more informative (reducing the dimensity of the data space), and the second is to reduce data noise.

The method of principal components is usually performed using the decomposition of the covariance matrix into its eigenvalues. However, this can also be done using singular value decomposition (SVD) of the data matrix A.

Let's assume there is a data matrix A of size $n \times p$, where n is the number of samples and p is the number of variables. Such a matrix is normalized, that is, the average values of the columns are subtracted and are now equal to zero.

On the next step, the $n \times p$ covariance matrix C is defined as $C = \frac{X^T X}{n-1}$. This is a symmetric matrix, so it can be diagonalized:

$$C = VLV^T,$$

where V is a matrix of eigenvectors (each column is an eigenvector) and L is a diagonal matrix with eigenvalues λ_i in decreasing order along the diagonal. The eigenvectors are called the principal axes of the data. Projections of data on the main axes are called the main components, or the weights of the main components; they can be considered as new, transformed variables. The j-th principal component is given by the j-th column of AV. The coordinates of the i-th data element in the new space of the principal components are given by the i-th line of AV.

If we now perform the distribution on the singular value of A, we will obtain such distribution:

$$A = USV^T,$$

where U is a unitary matrix and S is a diagonal matrix of singular values of s_i. From here it is easy to see that

$$C = VSU^T USV^T / (n-1) = V \frac{S^2}{n-1} V^T,$$

so, the right singular vectors of V are the principal directions, and the singular values are determined with the eigenvalues of the covariance matrix through $\lambda_i = s_i^2 / (n-1)$. The main components are set as follows:

$$AV = USV^T V = US.$$

PCA can also be formulated using approximate matrix factorization:

$A : N \times d$,
$N : Z \times k$,
$W : k \times d$,
$A \approx ZW$.

Singular Value Decomposition

The most common variant of LSA is based on the use of matrix decomposition with real elements by singular values or SVD decomposition (SVD – Singular Value Decomposition). With its help, any matrix can be decomposed into a set of orthogonal matrices, the linear combination of which is a fairly accurate approximation to the original matrix.

In linear algebra, the singular matrix decomposition is the factorization of this matrix into three matrices. It has some interesting algebraic properties and conveys important geometric and theoretical ideas about linear transformations. It also has some important applications in data science.

Singular matrix decomposition (SVD) is a powerful computational tool. Modern algorithms for obtaining such a decomposition of general matrices have had a profound impact on numerous applications in scientific and engineering disciplines. SVD is commonly used to solve unconstrained linear least squares problems, matrix rank estimation, and canonical correlation analysis. In computational science, it is commonly used in areas such as information retrieval, seismic reflection tomography, and real-time signal processing.

Let A be an $m \times n$ matrix and rank $A = r$. Therefore, the number of nonzero singular values of matrix A is equal to r. Since they are positive and labeled in decreasing order, we can write them as $\sigma_1 \geq \sigma_1 \geq \ldots \geq \sigma_n$ where $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$.

Each singular value of σ_i is the square root of λ_i (the eigenvalue of A^TA) and corresponds to the eigenvector v_i of the same order. Now we can write the singular decomposition of matrix A in the next form:

$$A = U \sum V^T,$$

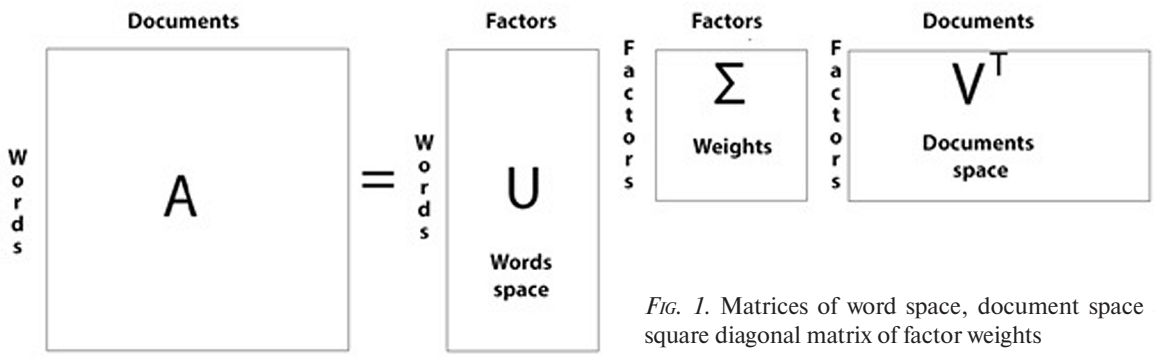

Fig. 1. Matrices of word space, document space and square diagonal matrix of factor weights

where V is an $n \times n$ matrix whose columns are v_i. Then:

$$V = [V_1 V_2 \dots V_n],$$

Σ is a diagonal $m \times n$ matrix of the form:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}.$$

U is an $m \times n$ orthogonal matrix.

If the term-document matrix is used as the source information, as a result of the singular decomposition, the matrix of the word space, the matrix of the document space and the square diagonal matrix of the factor weights will be obtained as shown in Fig. 1.

However, for many applications, the number r of nonzero singular values is too large, making even Compact SVD impractical for usage in applications. In such cases, it may be necessary to truncate the smallest singular values to compute only $t << r$ nonzero singular values. Truncated SVD is no longer an exact decomposition of the original matrix A, but rather provides an optimal approximation of the matrix of lower rank

$$\tilde{A} = U_t \Sigma_t V_t^*,$$

where the U_t is matrix of size $m \times t$, Σt is a $t \times t$ diagonal matrix, and V_t^* is a $t \times n$ matrix. Only t column-vectors of U and t row-vectors of V^* corresponding to the t largest singular values of Σ_t are calculated. It can be much faster and more economical than SVD if $t << r$.

Non-negative Matrix Factorization

Today, non-negative matrix factorization (NMF) is a very popular technology in artificial intelligence in general, and in computational linguistics in particular. Using of non-negative factorization within the paradigm of latent semantic analysis [6], computer linguists apply this approach to solving such applied problems as classification, clustering of texts and terms, building measures of semantic proximity, automatic selection of such linguistic structures from text corpora and relations, such as the preferences of conjugation in sentences (Selective Preferences) and subcategorical frames of verbs (Verb Sub-Categorization Frames), which include data on the semantic and syntactic properties of connections between verbs and their arguments – nouns in sentences, and many others. [8 –11].

Non-negative matrix factorization decomposes the non-negative matrix V into the product of the non-negative matrices W and H:

$$V \approx WH,$$

Where V is the matrix of size $m \times n$, $W - m \times k$ and $H - k \times n$, $k << n$, $k << m$ (Fig. 2.)

The distance measurement function between two non-negative matrices can be used as a metric function. One such metrics is the square of the Euclidean metric:

$$\mu = \|A - B\|^2 = \sum_{ij} (A_{ij} - B_{ij})^2.$$

Such a metric function is bounded from below. The lower bound 0 is reached if and only if $A = B$.

So, when using the Euclidean metric, matrix factorization goal is minimizing of $\|V - WH\|^2$ under the condition of non-negativity of W and H.

Such a metric function is non-increasing under the following rules [12]:

$$H_{ij} \leftarrow H_{ij} \frac{(W^T V)_{ij}}{(W^T WH)_{ij}}, \quad (1)$$

$$W_{ij} \leftarrow W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}}. \quad (2)$$

Execution of iterations of the algorithm continues until a stationary point is reached or the maximum number of iterations is executed.

At the same time, NMF has several advantages over other factors extraction methods in natural language processing. First, the matrices W and H have only non-negative elements, which simplifies their interpretation in terms of text understanding. Second, the columns of matrix W should not be orthogonal. Hence, the resulting topics can be directly interpreted, which seems to be quite common for real-world documents.

Computational Complexity of Algorithms

Let's consider the computational complexity of the truncated SVD for each step.

1. $O(mn^2)$ is required to calculate the matrix $A^T A \in R^{n \times n}$.

2. Execution of the own decomposition $A^T A \in R^{n \times n}$ requires $O(n^3)$.

3. Obtaining the square root of each eigenvalue of $A^T A$ requires $O(n)$.

4. Calculating $u_i = \frac{Av_i}{\sigma_i}$ takes $O(n(mn + m))$, since calculating Avi takes $O(mn)$, while dividing by σ_i takes $O(m)$. In total, we have n such equations, so $O(n(mn + m))$ is required.

The total computational complexity of the truncated SVD is $O(2mn^2 + mn + n^3 + n)$.

Also widely used is calculation for eigenvalues using the Lanzosh algorithm, the computational complexity of which is equal to $O(kn^2)$ for k iterations [13].

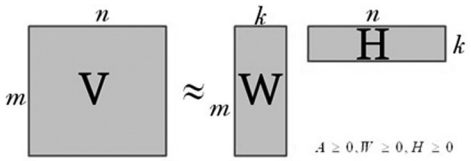

Fig. 2. Non-negative matrix factorization

To calculate the computational complexity of non-negative matrix factorization, formulas 1 end 2 can be reduced to their general form by making the replacement $H' = H^T$. We will get

$$(H'_t)_{ij} = (H'_{t-1})_{ij} \frac{(V^T W_{t-1})_{ij}}{(H'_{t-1} W_{t-1}^T W_{t-1})_{ij}}, \quad (3)$$

$$(W_t)_{ij} = (W_{t-1})_{ij} \frac{(VH'_t)_{ij}}{(W_{t-1} H_t'^T H'_t)_{ij}}. \quad (4)$$

Thus, we can present both formulas (3) and (4) in one form, thanks to the replacement of H', W and V^T, or W, H' and V by A, B and S, respectively:

$$A_{ij} = A_{ij} \frac{(SB)_{ij}}{(AB^T B)_{ij}}. \quad (5)$$

Formula (5) can be represented as a sequence of four steps (6, a –d):

$$C = SB, \quad (6, a)$$

$$K = B^T B, \quad (6, b)$$

$$D = AK, \quad (6, c)$$

$$A_{ij} = A_{ij} \frac{C_{ij}}{D_{ij}}. \quad (6, d)$$

These steps have a computational complexity of $O(k^*(nnz(S)+n))$, $O(k^2 m)$, $O(k^2 n)$, and $O(kn)$ respectively, where $nnz(S)$ is the number of non-zero elements matrix S.

Thus, the total computational complexity of NMF is only

$O(k^*(nnz(S)+k^2 m+k^2 n+2kn)$.

Performance Testing

A comparison of the performance of the truncated singular expansion and non-negative matrix factorization on dense and sparse matrices of different sizes was made. To calculate the eigenvalues for the singular decomposition, the Lanzosh algorithm [14] was used. Non-negative matrix factorization was performed using the multiplicative rules of Lee and Song [12]. Both algorithms were performed

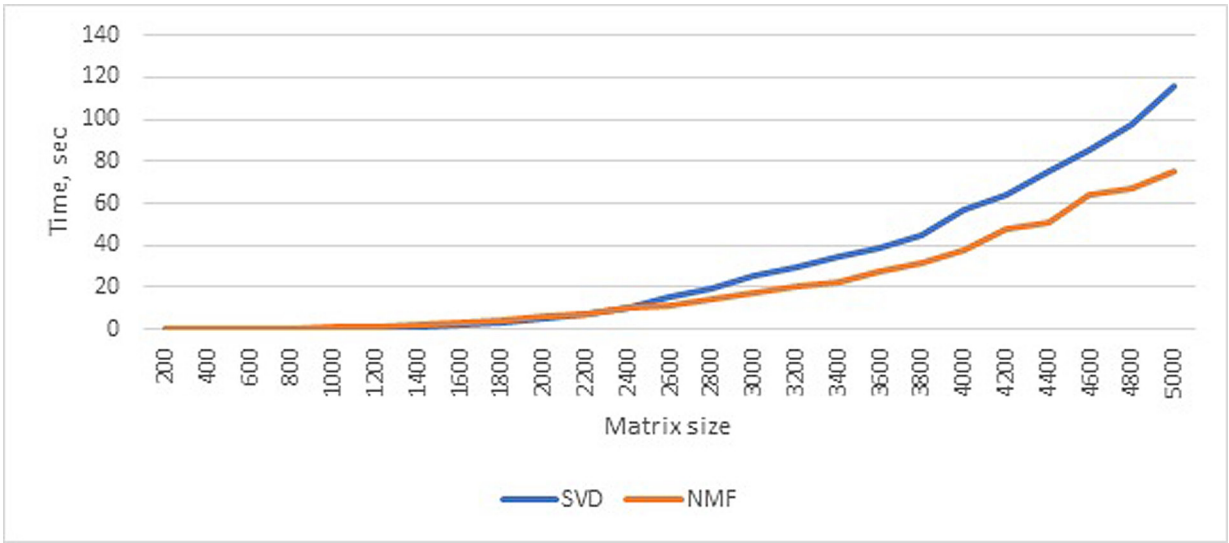

Fig. 3. Time of decomposition of dense matrices of different sizes

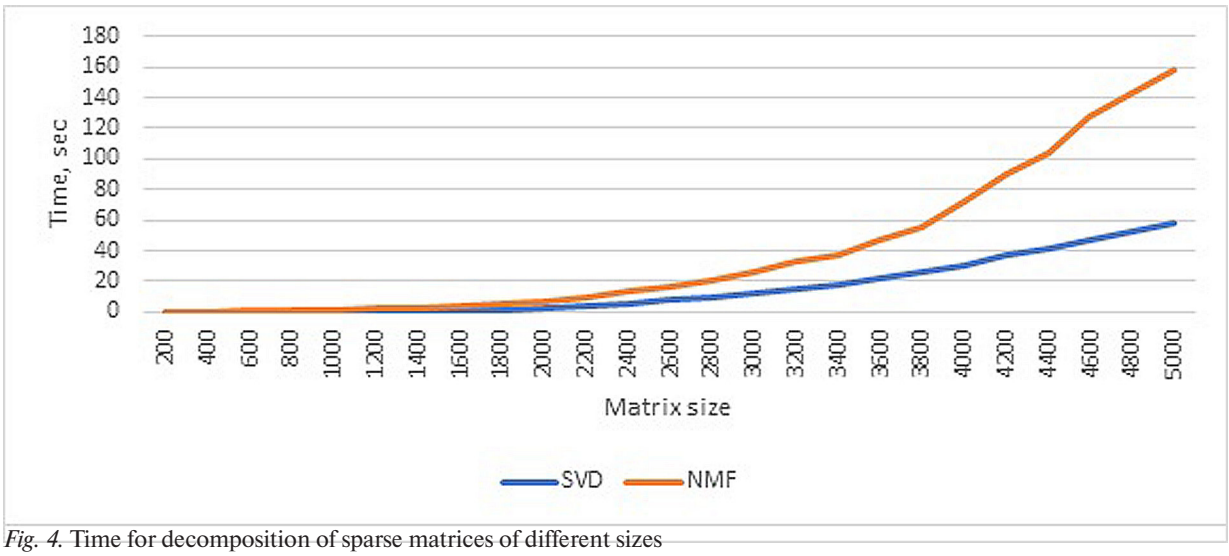

Fig. 4. Time for decomposition of sparse matrices of different sizes

with a limit of 200 iterations and a convergence rate of $1e^{-4}$.

Fig. 3, which shows the decomposition time of dense square matrices of various sizes, shows that non-negative matrix factorization takes less time with increase of matrix size. However, in the decomposition of sparse matrices (Fig. 4), the singular decomposition turned out to be faster than the basic algorithm of non-negative matrix factorization.

However, NMF, unlike truncated SVD, allows you to get results already after the first iterations, albeit with a larger error. Such data can be used and further updated, reducing the error. With truncated SVD, it is necessary to perform a certain number of iterations to obtain the required number of eigenvalues.

It is also worth noting that many parallel algorithms for processing sparse matrices have been developed for both methods. At the same time, the block-diagonal algorithm of non-negative matrix factorization proposed in [15] makes it possible to significantly reduce the time of non-negative factorization and to supplement the previously obtained model with new data.

Using of Algorithms to Multidimensional Arrays

A further development of natural language models in computational linguistics is the transition from two-dimensional matrices to multi-dimensional arrays of data, such as linguistic Subject-Predicate-Appendix tensors containing frequencies of occurrences of word combinations by parts of speech. Usually, non-negative factorization of tensors is used to reduce the dimensions of linguistic tensors [16]. The main difficulties of non-negative factorization of such linguistic tensors are their large size and significant sparseness.

However, for certain applications, it is possible to expand tensors by layers in the matrix (Fig. 5) and their subsequent decomposition using singular value decomposition or non-negative matrix factorization. In this way, the matrices Subject-[Predicate-AppendixCombinations], Predicate-[Subject-Appendix Combinations], Appendix-[Subject-Predicate Combinations] will be obtained. Thus, based on the matrices obtained after the factorization of any of the above matrices, it is possible to restore the frequency of occurrences of the Subject-Predicate-Appendix combination from the initial tensor.

This approach can have additional application value in the construction of recommender systems, automatic continuation of text input, etc.

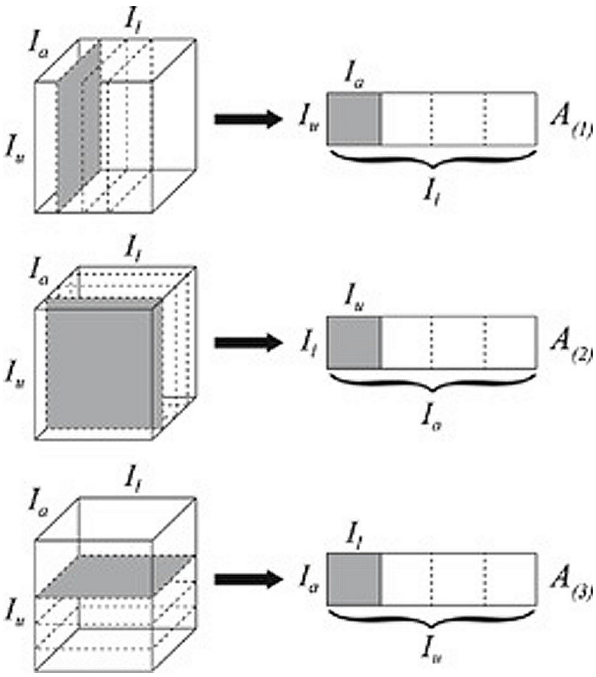

Fig. 5. Tensor expansion in a matrix

Conclusion

The non-negative factorization of matrices and the singular value decomposition of matrices are actually groups of algorithms. Both are commonly used for lower-rank factorization while minimizing the least-squares metric compared to the input data. The main difference is in the basis build by each algorithm.

With a singular value decomposition, an element from a data set is represented as a linear combination of vectors in the basis. So, there can be positive and negative coefficients, and they can have any value. The basis obtained by SVD helps to reconstruct the data by minimizing the metric, but received values are not easy to understand and do not have a specific meaning. In non-negative matrix factorization, the non-negativity constraint is imposed on the elements of the input and output matrices. In NMF, the coefficients are non-negative numbers, and then each vector in the basis is usually a small part used to reconstruct the elements.

The advantage of non-negative matrix factorization is twofold: first, it uses lower dimensionality, which inherently makes further computations much more efficient. But what differentiates this analysis from singular value decomposition is that, depending on what the input data represents, the W columns often capture meaningful characteristics of the data, and the H columns can be used to cluster the data in terms of those meaningful characteristics.

However, SVD performs a "deeper" factorization, as it outputs not only the U and V matrices, but also Σ. U and V are guaranteed to be an orthonormal basis, and Σ provides valuable information about the amount of information in each subsequent basis dimension. Such information is not available in the non-negative factorization of matrices.

REFERENCES

1. Allab, K., Labiod, L., Nadif, M., 2016. "A semi-NMF-PCA unified framework for data clustering". *IEEE Transactions on Knowledge and Data Engineering,* 29(1), pp. 2-16.
2. Kuang, D., Choo, J., Park, H., 2015. "Nonnegative matrix factorization for interactive topic modeling and doc-ument clustering". *Partitional clustering algorithms*, pp. 215-243.
3. Alghamdi, H., Selamat, A., 2015. "Topic modelling used to improve Arabic web pages clustering". In 2015 *International Conference on Cloud Computing* (ICCC, IEEE). pp. 1-6.
4. Hosseini-Asl, E., Zurada, J.M., 2014. "Nonnegative matrix factorization for document clustering: A survey". *In Artificial Intelligence and Soft Computing: 13th International Conference*, *ICAISC* 2014, Zakopane, Poland, June 1-5, 2014, Proceedings, Part II, LNAI 8468,, Springer International Publishing, pp. 726-737.
5. Klinczak, M.N., Kaestner, C.A., 2015. "A study on topics identification on Twitter using clustering algorithms". *In 2015 Latin America Congress on Computational Intelligence* (LA-CCI), pp. 1-6. IEEE.
6. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R., 1990. "Indexing by latent se-mantic analysis". *Journal of the American society for information science*, 41(6), pp. 391-407.
7. Dumais, S. T. (1991). "Improving the retrieval of information from external sources". *Behavior research methods, instruments, & computers*, 23(2), pp. 229-236.
8. Xu, W., Liu, X., Gong, Y., 2003. "Document clustering based on non-negative matrix factorization". In Pro-ceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp. 267-273.
9. Shahnaz, F., Berry, M. W., Pauca, V.P., Plemmons, R.J., 2006. "Document clustering using nonnegative matrix factorization". *Information Processing & Management*, 42(2), pp. 373-386. DOI: 10.1016/j.ipm.2004.11.005.
10. Van De Cruys, T., 2010. A Non-negative Tensor Factorization Model for Selectional Preference Induction. *Journal of Natural Language Engineering,* T. 16(4), pp. 417–437.
11. Van de Cruys, T., Rimell, L., Poibeau, T., Korhonen, A., 2012. "Multi-way Tensor Factorization for Unsuper-vised Lexical Acquisition". *Proceedings of COLING*-2012, pp. 2703–2720.
12. Lee, D.D., Seung, H.S., 2000. "Algorithms for non-negative matrix factorization". *In NIPS, MIT Press*, pp. 556–562.
13. Li, X., Wang, S., Cai, Y., 2019. Tutorial: Complexity analysis of singular value decomposition and its variants. https://arxiv.org/abs/1906.12085.
14. Golub, Gene H.; Van Loan, Charles F. "Lanczos Methods". Matrix Computations. *Baltimore: Johns Hopkins University Press*. 1996, pp. 470–507.
15. Marchenko, O.O., Nasirov, E.M., 2021. "Block-Diagonal approach for non-negative factorization of huge sparse linguistic matrices" *Proceedings Actual problems of theory of control systems in compures sciences,* pp. 72–78 (In Ukrainian).
16. Phan, A.H., Cichocki, A., 2008. "Multi-way Nonnegative Tensor Factorization Using Fast Hierarchical Alter-nating Least Squares Algorithm (HALS)". *In Proceedings 2008 International Symposium on Nonlinear Theory and its Applications*, pp.41–44.

Received 06.09.2023

ЛІТЕРАТУРА

1. *Allab K., Labiod L., Nadif M.* A semi-NMF-PCA unified framework for data clustering. IEEE Trans. Knowledge Data Eng. 2017. 29 (1). С. 2–16.
2. *Kuang D., Choo J., Park H.* Nonnegative matrix factorization for interactive topic modeling and document clustering. In: Partitional Clustering Algorithms. Springer; 2015. С. 215–43.
3. *Alghamdi H., Selamat A.* Topic Modelling used to improve Arabic Web pages Clustering. International Con-ference on Cloud Computing (ICCC); 2015. С. 1–6.
4. *Hosseini-Asl E., Zurada J.M.* Nonnegative matrix factorization for document clustering: a survey. In: Rutkow-ski L, editor. ICAISC 2014, Part II, LNAI 8468. С. 726–37.
5. *Klinczak M.N., Kaestner Celso A.A.* A Study on Topics Identification on Twitter using Clustering Algorithms. Latin America Congress on Computational Intelligence (LA-CCI); 2015. С. 1–6.
6. *Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.* Indexing By Latent Semantic Analysis. Journal of the American Society For Information Science. 41. 1990. С. 391–407.
7. *Dumais, S.T.* Improving the retrieval of infonnation from external sources. Behavior Research Methods, In-struments and Computers. 1990. 23, 229–236.
8. *Xu W., Liu X., Gong Y.* Document-clustering based on n-egative matrix factorization. In Proceedings of SIGIR'2003. С. 267–273.

9. *Shahnaz F.* et al. Document clustering using nonnegative matrix factorization. Information Processing and Management. Vol 42. 2006. С. 649–660.
10. *Van De Cruys T.* A Non-negative Tensor Factorization Model for Selectional Preference Induction. *Journal of Natural Language Engineering*, 2010. Т. 16(4). С. 417–437.
11. *Van De Cruys T.* et al. Multi-way Tensor Factorization for Unsupervised Lexical Acquisition. *Proceedings of COLING*-2012. 2012. С. 2703–2720.
12. *Lee D.D., Seung H.S.* Algorithms for non-negative matrix factorization. *In NIPS, MIT Press.* 2000. С. 556–562.
13. *Xiaocan L., Shuo W., Yinghao C.* Complexity analysis of Singular Value Decomposition and its variants. Uni-versity of Chinese Academy of Sciences. 2019. С. 1–12.
14. *Golub, Gene H.; Van Loan, Charles F.* "Lanczos Methods". Matrix Computations. Baltimore: Johns Hopkins University Press. 1996. С. 470–507.
15. *Марченко О.О., Насіров Е.М.* Блочно-діагональний підхід до невід'ємної факторизації розріджених лінгвістичних матриць надвеликої розмірності» Актуальні проблеми теорії керуючих систем у комп'ютерних науках: праці науково-практичної конференції. 2021. С. 72–78.
16. *Phan A.H., Cichocki A.* Multi-way Nonnegative Tensor Factorization Using Fast Hierarchical Alternating Least Squares Algorithm (HALS). In Proceedings 2008 International Symposium on Nonlinear Theory and its Applications. 2008. С. 41–44.

Надійшла 06.09.2023

О.О. Марченко, доктор фіз.-мат. наук, професор, зав. відділом, Міжнародний науково-навчальний центр інформаційних технологій і систем НАН та МОН України,
ORCID: https://orcid.org/0000-0002-5408-5279,
03187, м. Київ, просп. Академіка Глушкова, 40, Україна.
omarchenko@univ.kiev.ua
Е.М. Насіров, кандидат фіз.-мат. наук, старший науковий . співробітник, Міжнародний науково-навчальний центр інформаційних технологій і систем НАН та МОН України,
ORCID: https://orcid.org/0009-0006-9016-2602,
03187, м. Київ, просп. Академіка Глушкова, 40, Україна.
enasirov@gmail.com

МЕТОДИ ЗМЕНШЕННЯ РОЗМІРНОСТІ В АЛГОРИТМАХ ОБРОБКИ ТЕКСТІВ

Вступ. Методи зменшення розмірності є дуже важливим кроком у процесі кластеризації текстів. Вирішення такої задачі є складним через велику розмірність даних. Такий простір із великими розмірами знижує ефективність процесу загалом. Ідея цих методів полягає в тому, щоб зменшити розміри наявних об'єктів, перетворивши їх на новий простір із низькорозмірними об'єктами. Сучасними та широко вживаними методами зменшення розмірності є такі методи, як *NMF* та *SVD*.

Мета статті. Дослідити переваги та недоліки методів зменшення розмірності для застосування в алгоритмах обробки текстів, порівняти обчислювальну складність та швидкодію доліджуваних методів.

Методи. Системний підхід, аналіз, теорія складності обчислень.

Результати. Здійснено порівняння швидкодії усіченого сингулярного розкладу та невід'ємної матричної факторизації на щільних і розріджених матрицях різних розмірів. За результатами тестувань визначено, що невід'ємна матрична факторизація потребує менше часу для розкладу щільних квадратних матриць. Проте при розкладі розріджених матриць сингулярний розклад виявився швидшим ніж базовий алгоритм невід'ємної матричної факторизації.

Висновки. Невід'ємна факторизація матриць та сингулярний розклад матриць насправді об'єднують групами алгоритмів. Обидва зазвичай використовуються для факторизації нижчого рангу, при цьому мінімізують помилку найменших квадратів порівняно з вхідними даними. Основна відмінність полягає в тому, який базис будує кожен алгоритм.

Ключові слова: *штучний інтелект, комп'ютерна лінгвістика, паралельні обчислення.*