KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express ▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/icte

A probabilistic matrix factorization algorithm for approximation of sparse matrices in natural language processing

Gianmaria Tarantino*, Stefania Monica, Federico Bergenti

Dipartimento di Scienze Matematiche, Fisiche e Informatiche Universitá degli Studi di Parma, 43124 Parma, Italy

Received 12 February 2018; accepted 10 April 2018
Available online xxxx

Abstract

This paper suggests a variation of a well-known probabilistic matrix factorization algorithm which is commonly used in data analysis and scientific computing, and which has been considered recently to serve natural language processing. The proposed variation is meant to take benefit from the fact that matrices processed in natural language processing tasks are normally sparse rectangular matrices with one dimension much larger than the other, and this can be used to ensure adequate accuracy with acceptable computation time. Preliminary experiments on real-world textual corpora show that the proposed algorithm achieves relevant improvements compared to the original one.

Keywords: Latent semantic analysis; Natural language processing; Singular value decomposition

1. Introduction

The representation of words and documents as dense vectors is a fundamental step for several *NLP* (*Natural Language Processing*) tasks including information retrieval, word sense disambiguation, and text similarity. The literature proposes two types of methods to compute representations of words and documents [1]: *global matrix factorization methods*, and *local context window methods*. However, as discussed in [2], the distinction between such types of methods is becoming blurry since the well known method *skip gram with negative sampling* [3] implicitly factorizes a word-context matrix.

Among the global matrix factorization methods, a classic method to build dense representations of documents and words is *LSA* (*Latent Semantic Analysis*) [4]. Briefly, LSA processes a given corpus of textual documents by first extracting a normalized vocabulary of terms. Then, it builds a *TDM* (*Term-Document Matrix*) [5] whose element in position (i, j) is the number of occurrences of term i in document j. Finally, LSA decomposes the produced TDM by means of *SVD* (*Singular Value Decomposition*) [6] to reduce the size of the matrices that need to be processed with no loss of relevant information. Given that TDMs for real world corpora can have thousands of rows, *truncated SVD* [6] is often used to further reduce the size of matrices. Given a TDM W, its truncated SVD with rank $k \leq rank\,(W)$ is built using three matrices U_k, Σ_k and V_k such that

$$W = U_k \Sigma_k V_k^T, \tag{1}$$

where only the largest k singular values of W are considered to form the diagonal matrix Σ_k, and the corresponding left and right singular vectors to form the unitary matrices U_k and V_k. The representation of W in terms of Σ_k for a given k is used to obtain representations of words and documents as dense k-dimensional vectors. The choice of k influences the accuracy of the approximation of the TDM W in terms of the corresponding Σ_k.

The application of SVD to real world problems, which require the processing of huge corpora with a good approximation, normally requires prohibitive computation time even when

* Corresponding author.
E-mail addresses: gianmaria.tarantino@unipr.it (G. Tarantino), stefania.monica@unipr.it (S. Monica), federico.bergenti@unipr.it (F. Bergenti).

Peer review under responsibility of The Korean Institute of Communications Information Sciences.

https://doi.org/10.1016/j.icte.2018.04.005

Please cite this article in press as: G. Tarantino, et al., A probabilistic matrix factorization algorithm for approximation of sparse matrices in natural language processing, ICT Express (2018), https://doi.org/10.1016/j.icte.2018.04.005.

large computing infrastructures are used. The literature proposes a number of algorithms to address this problem, among which *probabilistic methods* are currently preferred because they can take benefit from the features of modern computing infrastructures. This paper proposes a probabilistic algorithm to process real world TDMs with sufficient accuracy in a reasonable computation time called *MQRR* (*Mixed QR Randomized subspace iteration with direct SVD*). Such an algorithm uses the fact that TDMs are commonly sparse rectangular to reduce needed computation time with minor or no loss of accuracy. The proposed algorithm is a variant of the well known *RSIDSVD* (*Randomized Subspace Iteration with Direct SVD*) [7], and a comparison with the performance of the original algorithm is shown in Section 3. In detail, the preliminary experimental results discussed in Section 3 show that MQRR achieves better accuracy at a lower computation time than RSIDSVD.

This paper is organized as follows. Section 2 presents the proposed MQRR algorithm focusing on the problem of computing low-rank decompositions of sparse rectangular matrices. Section 3 shows a preliminary assessment of the performance of the proposed algorithm. Finally, Section 4 concludes the paper and outlines planned future developments of this line of research.

2. The proposed algorithm

Probabilistic algorithms are commonly used for low-rank matrix approximation of large matrices. A detailed overview of state of the art probabilistic algorithms to construct approximate matrix decompositions is presented in [7]. Briefly, given a $m \times n$ matrix A for which a low-rank approximation is required, the basic idea of probabilistic low-rank approximation algorithms is to perform the following two steps [7]:

1. Compute an orthonormal matrix Q whose range approximates the range of A; and
2. Compute an approximate SVD factorization of A starting from $B = Q^T A$.

In detail, the first step is performed by factorizing matrix $A\Omega$ using QR decomposition [6], where Ω is a suitable $n \times l$ random matrix with Gaussian distribution, and l is chosen according to accepted approximation accuracy and computation time. The second step is performed by first factorizing B as $\tilde{U}\Sigma V^T$, and then computing the requested approximated SVD of A as $U\Sigma V^T$ where $U = Q\tilde{U}$.

The outlined scheme for probabilistic low-rank approximation can be improved for matrices whose singular values decay slowly, as assumed in the mentioned RSIDSVD. The idea of this technique is to apply randomized sampling to matrix $\tilde{A}$, defined in Proposition 1, for a small integer q, since the following proposition holds [7]:

Proposition 1. *Given a $m \times n$ real matrix A and defined $\tilde{A} := \left(AA^T\right)^q A$, $\tilde{A}$ has the same singular vectors of A and the following condition holds:*

$$\forall j \in [1..rank\,(A)] \quad \sigma_j\left(\tilde{A}\right) = \sigma_j(A)^{2q+1} \tag{2}$$

Algorithm 1 The pseudo-code of the proposed algorithm MQRR to perform an approximate SVD of an input matrix A.

1: **function** MQRR(A, l)
2: **input** $A : m \times n$ real matrix
3: **input** l : integer
4: **output** (U, Σ, V): U, V unitary, Σ diagonal
5: $\Omega \in \mathcal{N}^{n \times l}$
6: $Y_0 \leftarrow A\Omega$
7: $Q_0 R_0 = \mathrm{QR}_e(Y_0)$
8: **for** $j \leftarrow 1$ **to** q **do**
9: $\tilde{Y}_j \leftarrow A^T Q_{j-1}$
10: $\tilde{Q}_j \tilde{R}_j = \mathrm{QR}(\tilde{Y}_j)$
11: $Y_j \leftarrow A\tilde{Q}_j$
12: $Q_j R_j = \mathrm{QR}_e(Y_j)$
13: **end for**
14: $Q \leftarrow Q_j$
15: $B \leftarrow Q^T A$
16: $\tilde{U}\Sigma V^T = \mathrm{SVD}_e(B)$
17: $U \leftarrow Q\tilde{U}$
18: **return** (U, Σ, V)
19: **end function**

where $[a..b]$ denotes the set of integers between a and b, range boundaries included, and $\sigma_j(\cdot)$ denotes the jth singular value of a matrix, counted in descending order.

Note that [8] documents that in many cases a value of $q = 1$ or $q = 2$ is sufficient. Finally, note that randomized sampling applied to matrix $\tilde{A}$ is subject to rounding errors when executed using floating-point arithmetic, so it is necessary to orthonormalize the columns of the sample matrix after the applications of A and A^T.

The proposed algorithm focuses on improving the accuracy of RSIDSVD when dealing with sparse rectangular matrices. The pseudocode of the algorithm is shown in Algorithm 1, and it follows the general scheme of RSIDSVD outlined previously. The description of the algorithm assumes the following conventions. First, $\mathcal{N}^{n \times l}$ is used to denote the space of $n \times l$ random matrices with standard Gaussian distribution. Then, $\mathrm{SVD}_e(X)$ is used to denote the truncated SVD of matrix X at $rank(X)$, which from now on will be referred to as economy-size SVD. Finally, $\mathrm{QR}_e(Y)$ is used to denote the economy-size QR factorization of the $m \times n$ matrix Y with $m > n$, which is a QR decomposition of Y that uses only the first n columns of Q and the first n rows of R.

Note that the proposed algorithm uses a full QR factorization only at line 10, while the economy-size QR factorization is used at lines 7 and 12. The importance of the full QR factorization at line 10 is motivated by the following result from [6], and by the values of the singular values of common TDMs, as exemplified in next section.

Proposition 2. *Given a generic $m \times n$ matrix W, the following condition holds for any $k \leq rank(W)$:*

$$\min_{W' \in \mathcal{M}_k} \|W - W'\| = \sigma_{k+1}(W) \tag{3}$$

Please cite this article in press as: G. Tarantino, et al., A probabilistic matrix factorization algorithm for approximation of sparse matrices in natural language processing, ICT Express (2018), https://doi.org/10.1016/j.icte.2018.04.005.

Table 1
Computation times, in seconds, for the execution of FQRR and MQRR on two TDMs as the parameter l varies. The last two lines correspond to the time spent for computing exact and economy-size SVD, respectively.

l	CRANFIELD		CISI	
	FQRR	MQRR	FQRR	MQRR
10	9.1 s	3.1 s	11.3 s	3.5 s
25	9.6 s	3.1 s	12.1 s	3.5 s
50	10.3 s	3.1 s	13.6 s	3.7 s
100	11.7 s	3.5 s	16.1 s	4.0 s
200	9.7 s	3.3 s	11.5 s	4.0 s
300	9.7 s	3.3 s	12.2 s	3.9 s
SVD	4.0 s		4.8 s	
SVD_e	2.5 s		2.6 s	

Table 2
Spectral norm errors for the approximation of the input matrix using the factorization obtained by the execution of EQRR and MQRR on two TDMs as the parameter l varies. The last line corresponds to the spectral norm error obtained from approximating the input matrix with the exact SVD.

l	CRANFIELD		CISI	
	EQRR	MQRR	EQRR	MQRR
10	68.4	≤1.0e−11	44.6	≤1.0e−11
25	51.0	≤1.0e−11	33.8	≤1.0e−11
50	40.8	≤1.0e−11	28.1	≤1.0e−11
100	32.3	≤1.0e−11	22.0	≤1.0e−11
200	23.2	≤1.0e−11	16.9	≤1.0e−11
300	18.6	≤1.0e−11	14.0	≤1.0e−11
SVD	≤1.0e−11		≤1.0e−11	

where $\|\cdot\|$ is the spectral norm and $\mathcal{M}_k$ is the set of matrices with rank k, and the condition is realized by W' obtained via the economy-size SVD expressed in Eq. (1).

The application of at least one full QR decomposition in Algorithm 1 ensures that matrix B at line 15 is a $n \times n$ matrix, and therefore the economy-size SVD factorization at line 16 can lead to an approximation of the input matrix whose error is not bounded by the $l+1$ singular value of matrix W. Note that a full QR factorization could also be applied at line 7, instead that at line 10 but, under the assumption that $m > n$, it would result in a much slower computation. The reason is that q is a small integer, as observed previously, Y_0 is an $m \times l$ matrix and Y_j is a $n \times l$ matrix.

3. Experimental results

The proposed algorithm is tested on two TDMs,[1] obtained from standard information retrieval test collections[2] namely CRANFIELD (TDM 4563 × 1398) and CISI (TDM 5544 × 1460). The performance of the algorithm is compared with that of two variations of the RSIDSVD method. The first, which will be denoted as FQRR, is the RSIDSVD method in which all QR factorizations are full. The second, named here EQRR, is the RSIDSVD method in which all QR factorizations are economy-size. All experiments were performed calling the functions multiple times, then taking the median of the measurements. The results exposed in Table 1 are carried out using MATLAB function `timeit`. All experiments were performed using MATLAB R2017b on a laptop with an Intel(R) Core(TM) i7-3615QM 2.3 GHz processor and 8 GB of RAM. The MATLAB code used to perform experiments and to obtain results shown in Tables 1 and 2 can be downloaded at ailab.unipr.it/software/mqrr.zip.

Tables 1 and 2 report different executions of discussed algorithms for different values of l when input matrices are the TDMs obtained from CRANFIELD and CISI datasets. Each row of Table 1 shows the time, in seconds, spent for the execution of FQRR and MQRR algorithms, while Table 2 shows in its rows the spectral norm errors for the approximations of the input matrix with the computed factorization. Experimental results in Table 1 show that, for all tested values of parameter l, MQRR is computationally much more efficient than FQRR, maintaining the same order of accuracy in spectral norm than an exact SVD, as shown in Table 2. Moreover Table 1 shows that computation times for the execution of MQRR are less than exact SVD and comparable with economy-size SVD, both computed using the native commands that MATLAB provides. Finally, Table 2 shows that even small values of l, such as $l = 10$, give acceptable accuracy for the execution of MQRR. Small values of l can be preferred in order to obtain a slight execution speed up.

Better computational efficiency is gained from the execution of EQRR, but the large spectral norm errors reported in Table 2, even for large values of parameter l, prevent the use of such an algorithm when dealing with large rectangular sparse matrices. As outlined in Section 3, this behavior depends on the values of the singular values of the two TDMs. In fact, first note that all considered TDMs have their 301th singular value of order 10. But, for the case of the CISI TDM, all singular values up to the 1453th have at least order 1. Moreover, all singular values of the CRANFIELD TDM also have at least order 1. Therefore, even with larger value of l, the approximation obtained with EQRR cannot be comparable with that of the MQRR since Eq. (3) holds.

4. Conclusions

This paper proposes an algorithm to improve the performance of the construction of dense representations of words and documents from real world textual corpora. The TDMs of real world corpora are typically large, and sophisticated algorithms are needed to process them in an acceptable amount of time without introducing unacceptable approximations. The proposed algorithm, which is a variation of a well known algorithm, uses the fact that TDMs are normally rectangular sparse matrices to reduce the computation time and also to achieve better accuracy than the original algorithm. In addition, all matrix products in the proposed algorithm are intrinsically parallel, and it can be easily adapted to take benefit of the parallel computation features of modern computation infrastructures. Finally, a randomized blocked version of the economy-size QR factorization, as described in [8], can be used to modify the proposed algorithm to have much more benefits in terms of execution time with no loss of accuracy.

[1] http://scgroup20.ceid.upatras.gr:8000/tmg.
[2] http://web.eecs.utk.edu/research/lsi/corpa.html.

Please cite this article in press as: G. Tarantino, et al., A probabilistic matrix factorization algorithm for approximation of sparse matrices in natural language processing, ICT Express (2018), https://doi.org/10.1016/j.icte.2018.04.005.

Conflict of interest

The authors declare that there is no conflict of interest in this paper.

References

[1] J. Pennington, R. Socher, C.D. Manning, GloVe: Global vectors for word representation, in: Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1532–1543.

[2] O. Levy, Y. Goldberg, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Neural Word Embedding as Implicit Matrix Factorization, in: Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014, pp. 2177–2185.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., USA, 2013, pp. 3111–3119.

[4] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Amer. Soc. Inf. Sci. 41 (6) (1990) 391–407.

[5] G.W. O., M.W. Berry, S.T. Dumais, Using linear algebra for intelligent information retrieval, SIAM Rev. 37 (4) (1995) 573–595.

[6] G.H. Golub, C.F. van Loan, Matrix computations, in: Johns Hopkins Series in the Mathematical Sciences, vol. 3, The Johns Hopkins University Press, 1989.

[7] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.

[8] S. Voronin, P.-G. Martinsson, A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices, SIAM J. Sci. Comput. 38 (5) (2016) S485–S507.

Please cite this article in press as: G. Tarantino, et al., A probabilistic matrix factorization algorithm for approximation of sparse matrices in natural language processing, ICT Express (2018), https://doi.org/10.1016/j.icte.2018.04.005.