Available online at www.sciencedirect.com

ScienceDirect

ICT Express 4 (2018) 87–90

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 16 April 2018

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

$$W = U_k \Sigma_k V_k^T, \qquad (1)$$

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 and Information Sciences (KICS).

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:*

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

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.

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 $(U, \Sigma, V)$: $U, V$ unitary, $\Sigma$ 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 $(U, \Sigma, V)$
19: end function
```

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 $k \le rank(W)$:*

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