Using Matrix Decompositions in Formal Concept Analysis

Vaclav Snasel[1], Petr Gajdos[1], Hussam M. Dahwa Abdulla[1], Martin Polovincak[1]

[1] Dept. of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB-Technical University of Ostrava, 17. listopadu 15, Ostrava, 708 33, Czech Republic
{vaclav.snasel, petr.gajdos, hussamdahwa, martin.polovincak.fei}@vsb.cz

Abstract. One of the main problems connected with the formal concept analysis and lattice construction is the high complexity of algorithms which plays a significant role when computing all concepts from a huge incidence matrix. In some cases, we only need to compute some of them to test for common attributes. In our research we try to modify an incidence matrix using matrix decompositions, creating a new matrix with fewer dimensions as an input for some known algorithms for lattice construction. In this paper, we want to describe methods of matrix decompositions and their influence on the concept lattice.

Keywords: Formal Concept Analysis, Singular Value Decomposition, Semi Discrete Decomposition

1 Introduction

We are dealing with possible uses of matrix decompositions for reduction of concept lattice. These methods are well known in the area of information retrieval under the name Latent Semantic Indexing (LSI) or Latent Semantic Analysis (LSA) [2], [3]. LSI and LSA have been used for discovery of latent dependencies between terms (or documents). We would like to apply this approach in the area of formal concept analysis (FCA) [5]. First, we will introduce basic terms of Formal Concept Analysis (FCA) and then we will describe the rank-k SVD decomposition and the LSI method. Because the SVD decomposition presents some difficulty connected with the setting of its parameters, we will introduce a different approach based on semi-discrete lattice decomposition.

Small examples will illustrate our approaches. We hope that this is the way to use FCA on large data collections and visualize a data structure via concept lattice. Note that usage of this method depends on concrete application, because it makes some noise in the lattice structure to make it simpler.

2 Background

In the following paragraphs we would like to introduce important basic terms of formal concept analysis, singular value decomposition and semi-discrete value decomposition. Pros and cons of matrix decompositions will be discussed later.

2.1 Formal Concept Analysis

Definition 1. *[9] A formal context* $C = (G, M, I)$ *consists of two sets; G and M, I is a subset of GxM. The elements of* G *are defined as objects and the elements of* M *are defined as attributes of the context. In order to express that an object* $g \in G$ *is related to* I *with the attribute* $m \in M$ *, we record it as* gIm *or* $(g, m) \in I$ *and read it as "the object* g *has the attribute* m*".* I *is also defined as the context incidence relation.*

Definition 2. *[9] For* $A \subset G$ *as set of objects we use the definition* $A' = \{m \in M | gIm \text{ for all } g \in A\}$ *(the set of attributes common to the objects in A). Correspondingly, for B attributes we use the definition* $B' = \{g \in G | gIm \text{ for all } m \in B\}$ *(the set of objects which have all attributes in B).*

Definition 3. *[9] A formal concept of the context (G, M, I) is the pair (A, B) with* $A \subseteq G$*,* $B \subseteq M$*, and* $B' = A$*. We call A the extent and B the intent of the concept (A, B). β(G, M, I) denotes the set of all concepts of context (G, M, I).*

Definition 4. *[9] The concept lattice β (G,M, I) is a complete lattice in which infimum and supremum are given by*

$$\bigcap_{t \in T} (A_t, B_t) = (\bigcap_{t \in T} A_t, (\bigcup_{t \in T} B_t)'')$$
$$\bigcup_{t \in T} (A_t, B_t) = ((\bigcup_{t \in T} A_t)'', \bigcap_{t \in T} B_t)$$

2.2 Singular Value Decomposition

Singular value decomposition (SVD) is well known because of its application in information retrieval as LSI. SVD is especially suitable in its variant for sparse matrices. [3]

Theorem 1. *(Singular Value Decomposition) Let* A *be an* $m \times n$ *rank-r matrix. Be* $\sigma_1 \ldots \sigma_r$ *eigenvalues of a matrix* $\sqrt{AA^T}$*. There exist orthogonal matrices* $U = (u_1, \ldots, u_r)$ *and* $V = (v_1, \ldots, v_r)$*, whose column vectors are orthonormal, and diagonal matrix* $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$*. The decomposition* $A = U \Sigma V^T$ *is referred to as a singular value decomposition of matrix* A *and numbers* $\sigma_1 \ldots \sigma_r$ *are singular* *values of the matrix A. Columns of U (or V) are referred to as left (or right) singular vectors of matrix* A.

Now we have a decomposition of the original matrix *A*. Needless to say, the left and right singular vectors are not sparse. We have at most *r* nonzero singular numbers, where rank-*r* is the min(m,n). Because the singular values usually decrease quickly, we only need to take *k* greatest singular values and corresponding singular vector coordinates and create a *k*-reduced singular decomposition of matrix *A*.

Definition 5. *Let* us have k, 0 < k < r and singular value decomposition of A

$$A = U\Sigma V^T = (U_K U_0)\begin{pmatrix} \Sigma_K & 0 \\ 0 & \Sigma_0 \end{pmatrix}\begin{pmatrix} V_K^T \\ V_0^T \end{pmatrix}$$

$A = U_k \Sigma_k V_k^T$ is referred to as a *k*-reduced singular value decomposition (*k*-rank SVD).

In information retrieval, if every document is relevant to only one topic, we obtain a latent semantics – semantically related words and documents will have similar vectors in the reduced space. For an illustration of rank-*k* SVD see < 1, the grey areas determine first *k* coordinates from singular vectors, which are being used.

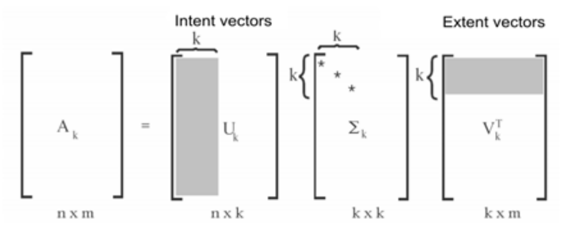

Fig. 1. k-reduced singular value decomposition

Theorem 2. *(Eckart-Young). Among all m×n matrices C of rank at most k, A_k is the one that minimizes* $\| A_k - A \|_F^2 = \sum_{i,j} (A_{i,j} - C_{w,j})^2$.

Because *rank-k* SVD is the best *rank-k* approximation of original matrix *A*, any other decomposition will increase the sum of squares of matrix $A - A_k$.

The SVD is hard to compute and once computed, it reflects only the decomposition of the original matrix. The recalculation of SVD is expensive, so it is impossible to recalculate SVD every time new rows or columns are inserted. The SVD-Updating is a partial solution, but since the error increases slightly with inserted rows and columns when updates occur frequently, the recalculation of SVD may be needed.

Note: *From now on, we will assume rank-k singular value decomposition when speaking about SVD.*

2.2 Semi Discrete Matrix Decomposition (SDD)

We have used another alternative decomposition in our research, semi discrete decomposition (SDD). For equal data set and incidence matrix, the SDD does as well as the SVD and uses less than one-tenth the storage space. In this section, we briefly overview SDD and then we will describe its purpose.

Semi discrete decomposition approximates a matrix as a weighted sum of outer product formed by vectors with entries constrained to be in the set $S = \{-1, 0, 1\}$. O'Leary and Peleg introduced the SDD in the context of image compression [8], and Kolda and O'Leary used the SDD for latent semantic indexing LSI in information retrieval [6], [7].

The primary advantage of the SDD over other types of matrix decompositions such as the truncated SVD is that it typically provides a more accurate approximation for far less storage. An SDD of an $m \times n$ matrix A as a decomposition is shown in figure 2. Here each x_i is an $m -$ vector with entries from the set S = {−1, 0, 1}, each y_i

$$A_k = [x_1 x_2 ... x_k] \begin{bmatrix} d_1 & 0 & ... & 0 \\ 0 & d_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & d_k \end{bmatrix} \begin{bmatrix} Y_1^T \\ Y_2^T \\ ... \\ Y_k^T \end{bmatrix}$$

Fig. 2. *k*-reduced semi discrete decomposition

is an $n-$vector with entries from the set S, and each d_i is a positive scalar. We call this $k -$ term SDD.

Note: *From now on, we will assume rank-k semi discrete decomposition when speaking about SDD.*

2.4 Nonnegative Matrix Factorization (NMF)

Nonnegative matrix factorization differs from other rank reduction methods for vector space models in text mining, e.g. principal component analysis (PCA) or vector quantization (VQ) due to use of constraints that produce nonnegative basis vectors, which make possible the concept of a parts-based representation.

Lee and Seung [11] first introduced the notion of parts-based representations for problems in image analysis or text mining that occupies nonnegative subspaces in a vector-space model. Techniques like PCA and VQ also generate basis vectors various additive and subtractive combinations of which can be used to reconstruct the original space. But the basis vectors for PCA and VQ contain negative entries and cannot be directly related to the original vector space to derive meaningful interpretations. In the case of NMF, the basis vectors contain no negative entries, allowing only additive combinations of the vectors to reproduce the original. So the perception of the whole, be it an image or a document in a collection, becomes a combination of its parts represented by these basis vectors. In text mining, the vectors represent or identify semantic features, i.e. a set of words denoting a particular concept or topic. If a document is viewed as a combination of basis vectors, then it can be categorized as belonging to the topic represented by its principal vector. Thus, NMF can be used to organize text collections into partitioned structures or clusters directly derived from the nonnegative factors [10].

3 Our Experiment

3.1 Input Data

Data from the Ostrava Mass Transit Authority was used in this experiment. We can describe data in a graph. Rows represent tram stops, columns represent tram lines. Let 1 represent a tram stopping at a tram stop in the appropriate row and column crossing, otherwise zero. This data was chosen because we can easily see if the SVD is working. Many tram lines share the same tram stops. These parts of tram lines should be minimized with SVD and it should also be visible in the lattice after using SVD.

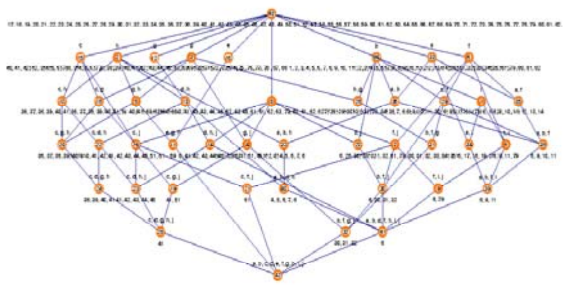

Fig. 3. Original lattice

3.2 Steps of Our Experiment

Our experiment included several steps. The first step was to read and transform data into adjacency matrix. After transforming and computing (using SVD) three new matrixes were obtained as results from SVD $A = U \Sigma V^T$. After choosing *rank-k*, a new matrix was computed.

3.3 Reading and Transforming Data

Data was obtained from tram timetables. The first ten tram lines and their tram stops were chosen. The transformed matrix had 92 rows and 10 columns. Transformation was simple. If a tram stops at a particular tram stop, the number 1 appeared in the appropriate row and column crossing, otherwise zero. The matrix is also an incidence matrix and was used in SVD, NMF and FCA computations.

3.4 Using SVD

SVD computations were made by SVDLIBC-fix software. This software is free and is written in C language. An incidence matrix acts as the input. Software can compute all three matrixes - U and Σ and V^T. k-rank was chosen $k = 4$, that is mean, matrixes will be U(92x10), Σ(10x10) and V^T(10x10).

3.5 Computing Concept Lattices

Both incidence matrix and product of SVD computing were used to compute concept lattice. Both matrixes passed as input for Concept Lattice software developed at VSB–TU Ostrava. The Concept Lattice software produced a concept lattice. The view of these concept lattices were completed with GraphPlace software. GraphPlace's output is a Postscript file that can be viewed in any postscript viewer.

Concept lattice was computed from the output matrix. Man can see the difference between the concept lattice computed from the original matrix and the concept lattice computed from the output matrix using SVD.

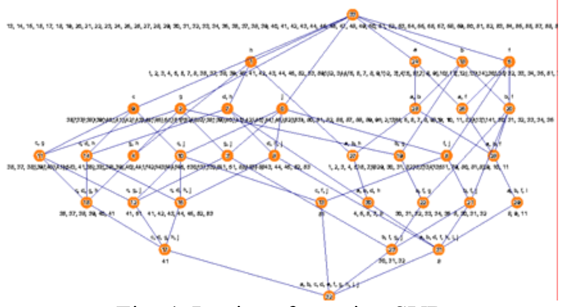
Fig. 4. Lattice after using SVD

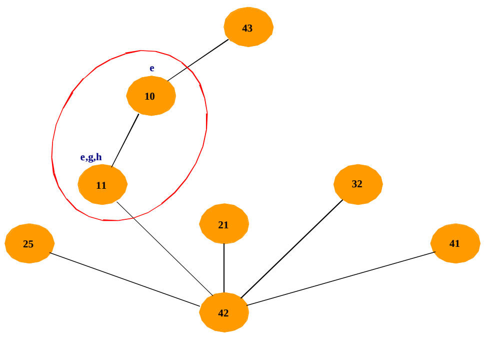

Fig. 5. Example 1

Example 1. Node 11 (figure 5): Node 11 has attributes (e, g, h) in line 69 of the original matrix. Attribute (e) disappears from line 69 after SVD. Attribute (e) is a shared attribute and exists only in this node at this level. When an attribute is lost the connection between node 11 and node 10 is lost too. The other attributes in node 11

(g, h) appear in the other nodes at the same level. For that node 11 is lost with node 10, which has lost all connections with other nodes.
Example 3. Node 34 (figure 7): Node 34 has attributes (a, i) in lines (8, 9, 11, 16, 17, 18, 19, 20, 21). After the SVD attribute (i) from lines (16, 17, 18, 19, 20, 21) is lost and only attribute (a) remains. Attribute (a) appears in other nodes at the same level (node 38), and lines (8, 9, 11) appear in other nodes at the same level (node 38).

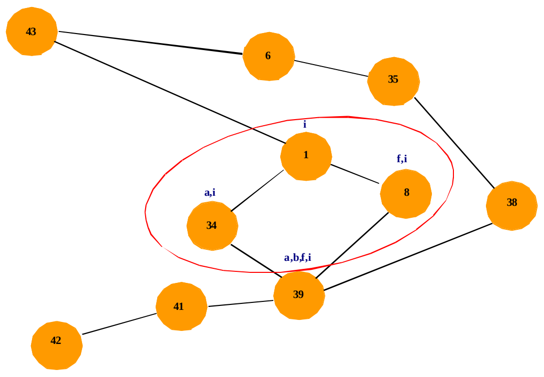

Fig. 6. Example 3

Also, node 8 has attributes (f, i) in lines (8,9,11,79), after SVD we lose the attribute (i) from line (79) and only attribute (f) remains, that appears in other node at the same level (node 38), and lines (8,9,11) appear in other node at the same level (node 38). For these nodes 34 and 8 are after SVD lost. Loosing nodes (34 and 8) means also loosing connection with node 1 who has attribute (i), and we loose also node 1.

5. Conclusion

Every node in the concept has a distinct set of attributes. That means that two nodes can not be found with the same set of attributes. Experiment consists in finding changes of objects and their set of attributes after using SVD. SVD reduces some attributes from the original matrix because these attributes do not appear in many objects. Attributes are not primary and they can be lost without any problem.

References

[1] 1. G. Battista, P. Eades, R. Tamassia, and I. Tollis, “Algorithms for drawing graphs, an annotated bibiliography”, Computational Geometry, Theory and Applications, vol. 4, 1994, pp. 235–282.
[2] M. Berry and M. Browne. “Understanding Search Engines”, Mathematical Modeling and Text Retrieval, Siam, 1999.

[3] M. Berry, S. Dumais, and T.Letsche. “Computation Methods for Intelligent Information Access”, Proceedings of the 1995 ACM/IEEE Supercomputing Conference, 1995.

[4] P.Gajdoš and P.Moravec, “Concept lattice generation by singular value decomposition”, CLA 2004, 2004, pp. 13–22.

[5] B. Ganter and R. Wille, Formal Concept Analysis. Springer-Verlag Berlin Heidelberg, 1999.

[6] T. G. Kolda and D. P. O’Leary, Computation and uses of the semidiscrete matrix decomposition, Technical Report CS-TR-4012, Oak Ridge National Laboratory, 1999, pp. 8-16.

[7] T.G.Kolda and D.P.O’Leary, “A semidiscrete matrix decomposition for latent semantic indexing information retrieval”, ACM Transactions on Information Systems 16(4), 1998, pp. 322–346.

[8] D.P.O’Leary and S.Peleg, “Digital image compression by outer product expansion”, IEEE Transactions on Communications 31, 1983, pp. 441–444.

[9] R. Wille, Lattices in data analysis: How to draw them with a computer, Algorithms and Order, 1989, pp. 33–58.

[10] Farial Shahnaz, Michael W. Berry, V. Paul Pauca, Robert J. Plemmons, Document clustering using nonnegative matrix factorization. Inf. Process. Manage. 42(2): 373-386 2006.

[11] Lee, D., Seung, “H. Algorithms for non-negative matrix factorization”, Advances in Neural Information Processing Systems, 2001, pp. 556-562.

[12] G.Young and C.Eckart, The approximation of one matrix by another of lower rank, Psychometrika 1, 1936, pp. 211–218.