Developing Genetic Algorithms for Boolean Matrix Factorization

Václav Snášel, Jan Platoš, Pavel Krömer

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{vaclav.snasel, jan.platos, pavel.kromer.fei}@vsb.cz

Abstract. Matrix factorization or factor analysis is an important task helpful in the analysis of high dimensional real world data. There are several well known methods and algorithms for factorization of real data but many application areas including information retrieval, pattern recognition and data mining require processing of binary rather than real data. Unfortunately, the methods used for real matrix factorization fail in the latter case. In this paper we introduce background and initial version of Genetic Algorithm for binary matrix factorization.

Keywords: binary matrix factorization, genetic algorithms

1 Introduction

Many applications in computer and system science involve analysis of large scale and often high dimensional data. When dealing with such extensive information collections, it is usually very computationally expensive to perform some operations on the raw form of the data. Therefore, suitable methods approximating the data in lower dimensions or with lower rank are needed. In the following, we focus on the factorization of two-dimensional binary data (matrices, second order tensors).

The paper is structured as follows: first, a brief introduction to matrix factorization is given. In the following section, the basics of Evolutionary and Genetic Algorithms are presented. The rest of the paper brings description of Genetic Binary Matrix Factorization and summarizes performed computer experiments and conclusions drawn from them.

2 Matrix Factorization

Matrix factorization (or matrix decomposition) is an important task in data analysis and processing. A matrix factorization is the right-side matrix product in

$$A \approx F_1 \cdot F_2 \cdot \ldots \cdot F_k \tag{1}$$

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 61–70, ISBN 978-80-248-1746-0.

for the matrix A. The number of factor matrices depends usually on the requirements of given application area. Most often, $k = 2$ or $k = 3$. There are several matrix decomposition methods reducing data dimensions and simultaneously revealing structures hidden in the data. Such methods include Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), which is our subject of interest in this research.

2.1 Non-negative matrix factorization

Non-negative matrix factorization (NMF) [1, 7] is recently very popular unsupervised learning algorithm for efficient factorization of real matrices implementing the non-negativity constraint. NMF approximates real $m\times\ n$ matrix A as a product of two non-negative matrices W and H of the dimensions $m \times r$ and $r \times n$ respectively. Moreover, it applies that $r << m$ and $r << n$.

$$A \approx W \cdot H \tag{2}$$

There are several algorithms for NMF computation based on iterative minimization of given cost function [7]. The original NMF algorithm involved minimization of the Frobenius norm [10] defined by formulae (3).

$$\|A - WH\|_F^2 = \sum_{ij} |A_{ij} - (WH)_{ij}|^2 \tag{3}$$

Other investigated cost measures include square of the Euclidean distance between V and its approximation (4) or Kullback-Leibler divergence D (5). For every cost function, there are update rules (multiplicative or additive) applied iteratively in order to reduce the distance between original matrix V and its model [7, 10].

$$\|A - WH\|^2 = \sum_{ij} (A_{ij} - (WH)_{ij}) \tag{4}$$

$$D(A \parallel WH) = \sum_{ij} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}) \tag{5}$$

Promising recent NMF algorithms are based on Gradient Descent Methods (GDM) or, extending the GDM, on Alternating Least Square computation [1, 10]. NMF was reported to give good results in extracting features or concepts from processed data. Unfortunately, the common NMF algorithms excelling in NMF computation for real valued matrices are unsuitable for efficient factorization of binary matrices.

2.2 Boolean matrix factorization

Boolean matrix factorization (BMF) or Boolean factor analysis is the factorization of data sets in binary $(1, 0)$ alphabet based on Boolean algebra. Boolean

factor analysis is extremely important in computer applications since the natural data representation for computerized processing is binary. Binary factorization finds its application in data mining, information retrieval, pattern recognition, image processing or data compression [5].

The BMF can be defined in a similar manner as NMF [11, 12, 14]. Consider binary[1] matrix A of the dimension $m \times n$ as a Boolean product of two binary matrices W and H of the dimensions $m \times r$ and $r \times n$ respectively. Let r be a subject to $r << m$ and $r << n$. Then, BMF is searching for best W and Hthat approximate A:

$$A \approx W \otimes H \tag{6}$$

where $\otimes$ stands for Boolean matrix multiplication.

The appeal of BMF lies in the fact that computerized data are binary in its essence and BMF is intensively investigated.

Keprt [5, 6] introduced BMF algorithms based on formal concepts and blind search. Meeds [8] et al. presented BMF model for factorization of dyadic data, however, Meeds' decomposition features one non-binary (integer) factor.

3 Evolutionary Algorithms

Evolutionary algorithms (EA) are family of iterative stochastic search and optimization methods based on mimicking successful optimization strategies observed in nature [2–4, 9]. The essence of EAs lies in the emulation of Darwinian evolution utilizing the concepts of Mendelian inheritance for the use in computer science and applications [2]. Together with fuzzy sets, neural networks and fractals, evolutionary algorithms are among the fundamental members of the class of soft computing methods.

EA operate with population (also known as pool) of artificial individuals (referred often as items or chromosomes) encoding possible problem solutions. Encoded individuals are evaluated using objective function which assigns a fitness value to each individual. Fitness value represents the quality (ranking) of each individual as solution of given problem. Competing individuals search the problem domain towards optimal solution [4].

For the purpose of EAs, a proper encoding representing solutions of given problem as encoded chromosomes suitable for evolutionary search process, is necessary. Finding proper encoding is non-trivial problem dependent task affecting the performance and results of evolutionary search while solving given problem. The solutions might be encoded into binary strings, real vectors or more complex, often tree-like, hierarchical structures, depending on the needs of particular application.

The iterative phase of evolutionary search process starts with an initial population of individuals that can be generated randomly or seeded with potentially good solutions. Artificial evolution consists of iterative application of genetic

[1] Matrix A is binary iff $\forall ij : [a]_{ij} = 0 \vee [a]_{ij} = 1$

operators (selection, crossover, mutation), introducing to the algorithm evolutionary principles such as inheritance, survival of the fittest and random perturbations. Current population of problem solutions is modified with the aim to form new and hopefully better population to be used in next generation. Iterative evolution of problem solutions ends after satisfying specified termination criteria and especially the criterion of finding optimal solution. After terminating the search process, evolution winner is decoded and presented as the most optimal solution found.

EAs are successful general adaptable concept with good results in many areas. The class of evolutionary techniques consists of more particular algorithms having numerous variants, forged and tuned for specific problem domains. The family of evolutionary algorithms consists of genetic algorithms, genetic programming, evolutionary strategies and evolutionary programming.

3.1 Genetic algorithms

Genetic algorithms (GA) introduced by John Holland and extended by David Goldberg are wide applied and highly successful EA variant. Basic workflow of originally proposed standard generational GA is:

I. Define objective function
II. Encode initial population of possible solutions as fixed length binary strings and evaluate chromosomes in initial population using objective function
III. Create new population (evolutionary search for better solutions):
    a. Select suitable chromosomes for reproduction (parents)
    b. Apply crossover operator on parents with respect to crossover probability to produce new chromosomes (offspring)
    c. Apply mutation operator on offspring chromosomes with respect to mutation probability. Add newly constituted chromosomes to new population
    d. Until the size of new population is smaller than size of current population go back to a.
    e. Replace current population by new population
IV. Evaluate current population using objective function
V. Check termination criteria; if not satisfied go back to III.

There are variants of standard generational GA. The differences are mostly in particular selection, crossover, mutation and replacement strategy [4].

4 Genetic Binary Matrix Factorization

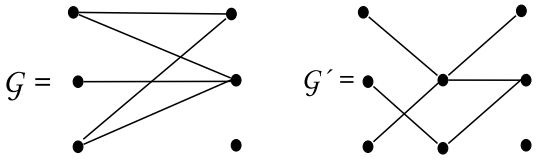
In this paper, we to propose a Genetic Algorithm for Binary Matrix Factorization (Genetic BMF - GBMF). For that, we first analzye the factors that are to be found by the algorithm and define an algorithm suggesting initial values of the factors.

4.1 Binary factors

The factors W and H found by NMF algorithm by Lee and Seung can be straightforwardly interpreted. Columns of W are basis vectors of column space of A and columns of H are weights associated with the base vectors. In order to find out interpretation of the matrix factorization task for GA which might be different, consider a graph-like representation of a matrix:

$$A = \begin{pmatrix} 1\,1\,0 \\ 0\,1\,0 \\ 1\,1\,0 \end{pmatrix} \equiv \mathcal{G} \approx WH = \begin{pmatrix} 1\,0 \\ 0\,1 \\ 0\,0 \end{pmatrix} \begin{pmatrix} 1\,1\,0 \\ 0\,1\,0 \end{pmatrix} \equiv \mathcal{G}' \tag{7}$$

Then, the factorization can be seen as a task of finding tripartite graph $\mathcal{G}'$ that will exclusively preserve the arcs between the pairs of 'edge' vertices from $\mathcal{G}$ in the form of a two step long paths through a 'middle layer'. Intuitively, the number of vertices in middle layer corresponds to r in NMF. The graphs $\mathcal{G}$ and $\mathcal{G}'$ from 7 can be depicted as follows:



When adopting this notion of Boolean matrix factorization, the interpretation of factors W and H slightly differs from the interpretation of NMF factors. The rows of H are base vectors of row space of A and rows of W are associated weights.

4.2 Constructive algorithm for suggesting base vectors of boolean matrix row space (CAS)

In order to provide the genetic algorithm with better than random initial population, a constructive algorithm for suggesting base vectors of matrix row space is defined:

I. Compute the cardinality (number of 1 elements) of the rows of A; divide rows of A to classes A^C per cardinality. Let W and H be 'empty'. Let $k = 0$.
II. Randomly pick up row A_i^C from the row class with lowest cardinality. Let $j = 0$.
III. For base row h_j in H :
   a. Check whether h_j is covered[2] by A_i^C. If not, attach A_i^C to H as h_k, and set $W[i,k] = 1$. Increase k and go back to II. In case h_j is covered by A_i^C, go to II.
   b. Let $W[i,j] = 1$. Let $A_i^C = A_i^C$ - w_j.
   c. If A_i^C is not zero vector, go back to II.

The rows of the matrix A can be then constructed using linear combinations of the rows of H.

[2] i.e. $h_j - A_i^C = o$ where $o = (0, 0, \ldots, 0)$ is zero vector.

4.3 Genetic algorithm for binary matrix factorization

We propose a genetic algorithm for BMF. It will exploit the initial factors constructed using the algorithm introduced in previous section. The objective function will be Hamming distance between reconstructed and original matrix. Crossover will aim to modify weights factor (matrix W) and mutation will primarily aim to alter basis vectors (matrix H). The algorithm can be summarized as follows:

I. Create initial population of N (WH) chromosomes and evaluate
II. Evolve population
   a. Select suitable chromosomes for reproduction (parents)
   b. Apply crossover on matrix W of selected parents.
   c. With very small probability, mutate W
   d. Mutate H
   e. Migrate offspring chromosomes to population
III. Evaluate current population using objective function
IV. Check termination criteria; if not satisfied go back to II.

The evaluation of chromosomes in population is implemented as comparison of original matrix V to the product of W and H. The fitness function is based on Euclidean distance between V and WH:

$$f = \frac{1}{\sqrt{\sum_i \sum_j (V[i,j] - WH[i,j])^2}} \tag{8}$$

The termination criteria were based on specified threshold defining minimum acceptance of evolved solution and maximum number of generations processed. The maximu number of generations was se to 1000 and the minimum acceptance 0.3.

In this way, the algorithm explores different combinations of base vectors (via crossover) and simultaneously adjust the base vector suggestions. Evolutionary principles will be applied and the factor interpretation maintained.

5 Experimental Algorithm Evaluation

This section provides summary on computer experiments conducted in order to verify proposed algorithms.

5.1 Evaluation of CAS algorithm

In 4.2 was defined a constructiove algorithm to suggest base vectors for initial matrix factors. We have compared the average error obtained after using random initial factors and initial factors suggested by CAS for several random square matrices. For every matrix dimension n, we have computed the average error (i.e. Hamming distance) between original matrix and its reconstruction. The

Table 1. Comparing average error of random initial factors and CAS suggested initial factors .

N	Random	CAS
32	513, 82	216
64	2047, 49	915
128	8195, 1	3809
256	32775, 3	15492
512	131070	63268

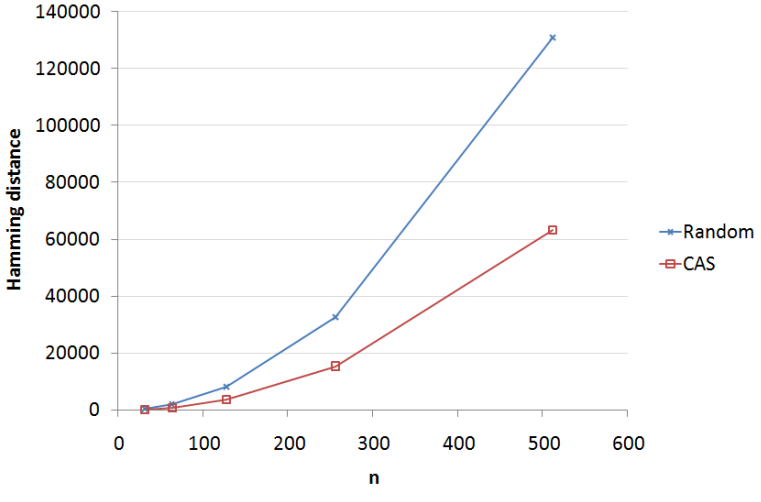


Fig. 1. The comparison of Random initial error and CAS initial error.

dimension was reduced to $\frac{n}{2}$ and the experiment was repeated 100 times for each n.

Obviously, the error produced by random initial factors is aproximately twice as big as the error obtained when using CAS initial factors.

5.2 Evaluation of GBMF algorithm

GBMF was implemented and run for some testing binary matrices. Black and white images were chosen as representation of input and output binary matrices for the ease of visual interpretatin of the results.

In all cases, GBMF was run with the following parameters: a population of 50 prospective factors, probability of crossover 0.9 and probability of mutation 0.2. GBMF was executed for 5000 generations.

The algorithm was first tested on a set of bar images. Testing bar images contained white background and black bars – vertical or horizontal lines gener-

ated with certain probability (0.2 for horizontal bar and 0.3 for vertical bar). The dimension of testing images was 15×25 pixels.

Initially, a small collection of 6 bar images (shown in figure 5.2) was processed by the algorithm to reduce its dimension to 3.

Fig. 2. Input bar images.

Fig. 3. Reconstructed bar images.

Fig. 4. Obtained base images.

Next, the progam was used to process in the same way a collection of 25 face images taken from the facial expression collection. The images were transformed from grayscale original to black and white (so they could be interpreted as binary matrices). The dimension of testing images was 19×19 pixels.

In both cases delivered the algorithm a set of images (i.e. matrices) which clearly share some elements with the original input. Albeit some base images were obtained, they did not contain distinguished features as for instance when using NMF for non-binary matrices. Moreover, the black and white images used for algorithm evaluation are not typical real world binary matrices. Also both binary and real-valued pseudorandom matrices do not contain features significant for matrices describing real world phenomena (i.e. in economics, physics etc.).

6 Conclusions and Future Work

In this paper, we have introduced initial version of a genetic algorithm for binary matrix factorization. In order to define Genetic Algorithm oriented approach to

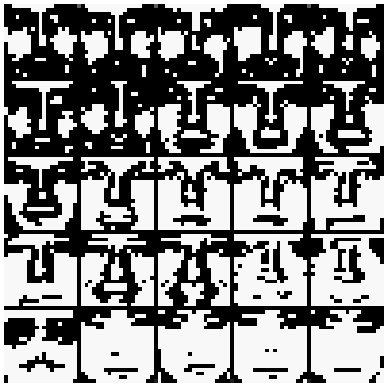


Fig. 5. Input face images.

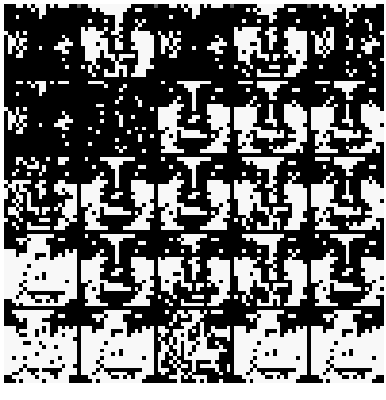


Fig. 6. Reconstructed face images.

Fig. 7. Obtained base faces.

BMF, an interpretation of binary factors was presented. Next, an algorithm for lossless BMF was used to create initial binary factors. Genetic BMF was defined, implemented and applied on first sample problems. In the future, we will focus on tuning the GBMF mplementation and evaluation of the algorithm on real world binary matrices. Some matrix properties such as sparsity might be exploited for algorithm modification.

References

1. Michael W. Berry, Murray Browne, Amy N. Langville, Paul V. Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization.
2. Ulrich Bodenhofer. *Genetic Algorithms: Theory and Applications*. Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter 2003/2004.
3. Mehrdad Dianati, Insop Song, and Mark Treiber. An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, July 2002.
4. Gareth Jones. Genetic and evolutionary algorithms. In Paul von Rague, editor, *Encyclopedia of Computational Chemistry*. John Wiley and Sons, 1998.
5. Ales Keprt. Using blind search and formal concepts for binary factor analysis. In Václav Snásel, Jaroslav Pokorný, and Karel Richta, editors, *DATESO*, volume 98 of *CEUR Workshop Proceedings*, pages 128–140. CEUR-WS.org, 2004.
6. Ales Keprt and Václav Snásel. Binary factor analysis with help of formal concepts. In Václav Snásel and Radim Belohlávek, editors, *CLA*, volume 110 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
7. Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.
8. Edward Meeds, Zoubin Ghahramani, Radford M. Neal, and Sam T. Roweis. Modeling dyadic data with binary latent factors. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 977–984. MIT Press, Cambridge, MA, 2007.
9. Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
10. Farial Shahnaz, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Inf. Process. Manage.*, 42(2):373–386, 2006.
11. Moravec, P., Snášel, V., Frolov, A. A., Húsek, D., Řezanková, H., Polyakov, P.: Image Analysis by Methods of Dimension Reduction. IEEE CISIM 2007: Elk, Poland, Pages 272–277
12. Snášel, V., Húsek, D., Frolov, A. A., Řezanková, H., Moravec, P., Polyakov P.: Bars Problem Solving - New Neural Network Method and Comparison. MICAI 2007: LNCS 4827 Springer 2007, Pages 671–682
13. Spellman, P.T., Sherlock, G., Zhang, M.Q., Anders, V.I.K., Eisen, M.B., Brown, P., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. In: Molecular Biology of the Cell 9. (1998) Pages 3273–3297
14. Zhang, Z., Tao Li, T., Ding, C., Zhang, Xiang-Sun: Binary Matrix Factorization with Applications. In Proceedings of 2007 IEEE International Conference on Data Mining (ICDM 2007),