Data Decomposition and Matrix Reduction using Binary-PSO in Tensor Mining

Mashkoor Ahmad[1] Sunil Kumar[2]
[1,2]Department of Compute Science and Engineering
[1,2]Pranveer Singh Institute of Technology, Kanpur, 209305, India

Abstract— Tensor decompositions are efficient method for big data analytics as it brings multiple modes and aspects of data to a unified framework, which allows us to discover complex internal structures and correlations of data. In existing approach are not designed to meet the major challenges posed by big data analytics, the distributed data correlation inspired from apriori algorithm which find out association rules that consider missing data afterwards it covers all local nodes in a network. Our proposed approach, attempts to improve the scalability of tensor decompositions and reduce missing data. The globally considered nodes which minimizing relative error , finding optimal solution then improving efficiency as with total number of frequent item-set using low rank approximation matrices for item-set weightage over nodes then assign fitness value as per global best function using LRA-PSO(low rank approximation based particle swarm optimization) approach for same association based distributed data using MATLAB 2014 Ra.

Key words: ALS, PSO, LRA Tensor Mining

I. INTRODUCTION

A tensor is a multidimensional array. More formally, an N-way or Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system. This concept of tensors is not to be disordered with tensors in physics and manufacturing (such as strain tensors) [5], which are generally referred to as tensor fields in mathematics [9]. A third-order tensor has three indices as shown in Figure 1. A 1st tensor is a vector; a 2nd tensor is a matrix and tensors of order three or higher is called higher-order tensors. The goal of this survey is to provide an overview of higher-order tensors and their decompositions. On the other hand there has been vigorous analysis on tensor decompositions and illustrations (i.e., decompositions applied to data arrays for extracting and explaining their properties) for four decades, very little of this work has been published in applied mathematics journals. Therefore, we wish to bring this research to the attention of SIAM readers.

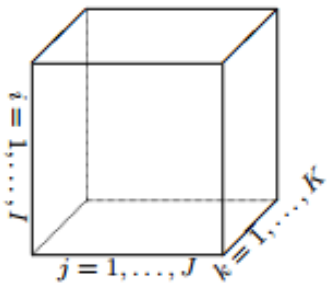

Fig. 1: A third-order tensor: $X \in^{R\ I\times J\times K}$

The N-way Toolbox for MATLAB, by Andersson and Bro [9], provides a large collection of algorithms for computing different tensor decompositions. It provides methods for computing CP and Tucker, as well as many of the other models such as multilinear partial least squares (PLS). Additionally, many of the methods can handle constraints (e.g., non-negativity) and missing data. CuBatch is a graphical user interface in MATLAB for the analysis of data that is built on top of the N-way Toolbox. Its focus is data centric, offering an environment for preprocessing data through diagnostic assessment, such as jack-knifing and bootstrapping. The interface allows custom extensions through an open architecture.

II. ALS (ALTERNATING LEAST SQUARE)

Tensor decomposition solves iteratively the below Equation(1) by an Alternating Least Squares algorithm which calculates concentration C and pure spectra S^T matrices that optimally fit the experimental data matrix D with non-modelled residuals E.

$$D = CS^T + E \quad (1)$$

This optimization is carried out for a proposed number of components and using original estimates of either C or S^T. In this work, initial estimates were determined using a pure variable method based on the Bisection approach [1]. ALS multistep analysis of multiple independent experiments run under different experimental conditions is a useful and powerful strategy to improve resolution. This strategy implies the analysis of a column-wise and row-wise data matrix, in which the resolved pure spectra of the same species are common for all experiments and experiment-to-experiment variation is allowed for the resolved concentration profiles. Runs combined in the multiset structure can be of different nature and size, e.g., runs from an experimental design, calibration and/or test runs, spectral information about pure compounds, etc. depending on the purpose of the analysis. Equation 2 shows the bilinear ALS model for a multiset system (D_1, D_2, ..., D_n), formed by nD_i experiments, which is expressed by a common pure spectra matrix S^T and sub-matrices of process profiles C_1, C_2, …, C_n related to D_1, D_2 … $D_n$respectively.

$$\begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{bmatrix} S^T + \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \quad (2)$$

Convergence during the optimization is achieved when in two consecutive iterative cycles, relative differences in standard deviations of the residuals between experimental and ALS calculated data values are less than a previously selected value, in this case 0.1%. Figures of merit of the optimization procedure are the percent of lack of fit (%LOF) and the percent of variance explained (%R2). Lack of fit is defined as the difference among the input data D and the data reproduced from the CS^T product obtained by ALS. This value is calculated according to the expression:

$$\%\, LOF = 100 \sqrt{\frac{\sum_{i,j} e_{ij}^2}{\sum_{i,j} d_{ij}^2}} \qquad (3)$$

where d_{ij} designs an element of the input data matrix D and e_{ij} is the related residual obtained from the difference between the input element and the ALS reproduction.

III. LOW-RANK APPROXIMATION OF A MATRIX

Given an MxN matrix C and a positive integer K, we wish to find an MxN matrix C_k of rank at most K, so as to minimize the *Frobenius norm* of the matrix difference X=C_k, defined to be using psooptimization :

$$\min_{\boldsymbol{u},\boldsymbol{v}} \left\| \boldsymbol{M} - \boldsymbol{u}\boldsymbol{v}' \right\|^2 + 2\lambda_1 \sum_{i=1}^{p} |u_i| + 2\lambda_2 \sum_{j=1}^{q} |v_j| \qquad (4)$$

When u is far smaller than r, we refer to M as a low-rank approximation.

The SVD is help to solve the low-rank matrix approximation difficulty. We then derive from it an application to approximating term-document matrices. We invoke the following three-step procedure to this end:

$\boldsymbol{t}_h = \boldsymbol{X}_{h-1}\boldsymbol{u} / \boldsymbol{u}'\boldsymbol{u}$ where tis the variable time

$\boldsymbol{w}_h = \boldsymbol{Y}_{h-1}\boldsymbol{v} / \boldsymbol{v}'\boldsymbol{v}$ where w is weight node vector of Y

$\boldsymbol{c}_h = \boldsymbol{X}_{h-1}^{\mathrm{T}}\boldsymbol{t}_h / \boldsymbol{t}_h'\boldsymbol{t}_h$ where c is constand vector of X

$\boldsymbol{d}_h = \boldsymbol{Y}_{h-1}^{\mathrm{T}}\boldsymbol{t}_h / \boldsymbol{t}_h'\boldsymbol{w}_h$ where minPts of Y

Thus, it seems reasonable that replacing these small eigenvalues by zero will not substantially alter the product, leaving it ``close" to c and d .

IV. PROPOSED METHOD

In our proposed approach consist of low ranking approximated to find missing detection rate with decomposed subset rules, Particle swarm optimization algorithm is an evolutionary computational technique where swarm describes the behavior for each node globally with fixed association rules. To get the optimum solution, this technique considers population based searching in corresponding to the eigenvalue with the maximum absolute value for distributed min-points where particles change their positions in a given space with respect to time. The particles are flying in a multidimensional space to find the solution in PSO for best rank-one matrix approximation of *M* is the creation of the first left and right singular vectors with minimal solution.

A. Proposed algorithm for low rank approximation for particle swarm optimization:

The proposed approach contains two parts:

1) The first part provides procedures related to encoding and calculating the fitness values after low-rank approximation of D with missing elements filled in with zeros of each particle in swarm.
2) In the second part of the approach, the particle swarm optimization algorithm is employed to mine the association rule set of indexes of the non-missing elements.
3) A multidimensional vector of D each particle considers two experiences to modify its current position. One is the best fitness value it has achieved called “pbest” and another is the best fitness value achieved by any particle of the generated population called “gbest”.
4) The approximated velocity of i-th particle at time t, then for calculating the new velocity of i-th particle at time t+1, which considers two best values pbest and gbest and it is

$v_i(t+1) = v_i(t) + r_1 (pbest_i – x_i (t) + r_2 (gbest_i – xi (t) \qquad (5)$

$x_i(t+1) = x_i (t) + v_i(t+1) \qquad (6)$

The pseudo code of proposed LRA-PSO approach shown in Figure 2, the algorithm is run as N times or number of desired rules. Moreover, each run includes a number of generations.

B. Pseudo code for proposed work(LRA-PSO)

Input: Database of Transaction (D)
Output: Best discovered association rules

1) t=0 // here t is iteration number
2) minPts← 0 , conf←0
3) Ck= gen(Lk−1) /Generates the candidate set as per iteration number
4) DiscoveredBestRules=∅ , count
5) initialize a population (M) particles consist with candidate set
6) increase the count by only 1 for each item appearance.
   f0(i) <=f(i) // where f0(i) function of particle count
7) While(t<N) do // algorithm run for number of iterations(N) and follow step 6.
8) begin
9) i=0 // here loop starts for each particle
10) repeat
11) Calculating the total approximated frequencies:
12) calculate fitness of each particle using approximate frequencies
13) Find size of lower approximated nodes for initial velocities.
14) For each particles (P~→ frequencies) do
15) begin
16) For each dimension d in P
17) begin
18) calculate empty stack velocity and position of each particle in d-dimension
19) Now find
    max(size(gbest),size(B)).*(size(pbest)>0&
    size(gbest)>0)
20) End for each low approximate dimension
21) End for each particle in fitting data
22) i++
23) compute set of indexes of the non-missing elements in the *i*th row *Di* .
24) *r* := *Di*,Pdiag(conf,P) = *Di*,P⊙S*i*,P // where r is relative maximal error for each node
25) Best ←Particle [Bestparticleindex]
26) until not terminate(i)
27) DiscoveredBestRules← DiscoveredBestRules ∪ Best for average approximated value
28) t++
29) End

At the beginning of the algorithm, the DiscoveredBestRulesset is empty. At the first iteration of algorithm, each particle is initialized randomly as a rule. In each of generation, until the reaching the termination

conditions, particles are evaluated and then calculate fitness of each particles. The individuals of population are sorted in descending order according to their fitness value. In each generation, the best discovered rule is added to Best vector and then for each particles update their velocity and position according to Eqs. (5) and (6). The rule is valid if it has at least one attribute in the antecedent of the rule and one in the consequence of the rule. At each iteration after reaching the termination condition best discovered rule from the Bestis added to the DiscoveredBestRul*e*sset. This process is continued until termination conditions not occur.

V. RESULT AND DISCUSSION

As there is no straightforward algorithm is known to determine the rank of a tensor (i.e. smallest number of components in data decomposition of the tensor, we used cross-validation to estimate the rank by trying different number of components during training. The proposed method significantly improves the imputation accuracy in terms of error rate compared to existing methods. Imputing missing data should not strongly affect the variance of the data. Obviously, mean substitution significantly reduces the variances, since it replaces all missing data by the mean value.

Rule #1: 5 --> 1 Support for Users = 0.22222 Confidence for Users = 1	Rule #2: 4 --> 2 Support for Users = 0.22222 Confidence for Users = 1	Rule #3: 5 --> 2 Support for Users = 0.22222 Confidence for Users = 1
Rule #4: 5 --> [1 2] Support for Users = 0.22222 Confidence for Users = 1	Rule #5: [1 5] --> 2 Support for Users = 0.22222 Confidence for Users = 1	Rule #6: [2 5] --> 1 Support for Users = 0.22222 Confidence for Users = 1
Rule #7: 1 --> 2 Support for Users = 0.44444 Confidence for Users = 0.66667	Rule #8: 1 --> 3 Support for Users = 0.44444 Confidence for Users = 0.66667	Rule #9: 3 --> 1 Support for Users = 0.44444 Confidence for Users = 0.66667
Rule #10: 3 --> 2 Support for Users = 0.44444 Confidence for Users = 0.66667	Rule #11: 2 --> 1 Support for Users = 0.44444 Confidence for Users = 0.57143	Rule #12: 2 --> 3 Support for Users = 0.44444 Confidence for Users = 0.57143
Rule #13: [1 2] --> 3 Support for Users = 0.22222 Confidence for Users = 0.5	Rule #14: [1 3] --> 2 Support for Users = 0.22222 Confidence for Users = 0.5	Rule #15: [2 3] --> 1 Support for Users = 0.22222 Confidence for Users = 0.5
Rule #16: [1 2] --> 5 Support for Users = 0.22222 Confidence for Users = 0.5	Rule #17: 1 --> 5 Support for Users = 0.22222 Confidence for Users = 0.33333	Rule #18: 1 --> [2 3] Support for Users = 0.22222 Confidence for Users = 0.33333
Rule #19: 3 --> [1 2] Support for Users = 0.22222 Confidence for Users = 0.33333	Rule #20: 1 --> [2 5] Support for Users = 0.22222 Confidence for Users = 0.33333	Rule #21: 2 --> 4 Support for Users = 0.22222 Confidence for Users = 0.28571
Rule #22: 2 --> 5 Support for Users = 0.22222 Confidence for Users = 0.28571	Rule #23: 2 --> [1 3] Support for Users = 0.22222 Confidence for Users = 0.28571	Rule #24: 2 --> [1 5] Support for Users = 0.22222 Confidence for Users = 0.28571

Table 1: Association rule mining of tensor data

Above table shows the using proposed method we get reduced calculation of data

In initial stage till rule 6 the value of Confidence for Users = 1

In second stage till 7-10 the value is Confidence for Users =0.66667

In third between 11-12 the value of Confidence for Users = 0.57143

In fourth between 13-16 the value of Confidence for Users = 0.5

In fifth between 17-20 the value of Confidence for Users = 0.33333

In six between 21-24 the value of Confidence for Users = 0.28571

Clearly we can see that the data calculation value in above description.

Association-ALS : res=10455.322038 res*=8107.896957. Runtime: 12.483302s. *** Proposed-PSO : res=10468.223163 res*=8104.277378. Runtime: 10.601163s. *** Proposed-PSO : res=10590.965591 res*=8151.464689. Runtime: 61.793788s. *** Association-ALS: res=10470.345410 res*=8123.765900. Runtime: 65.865101s. Elapsed time is 18.846886 seconds. Complete. Fit=0.978931

Table 2: Comparison between runtime and elapsed time

Above table shows the Association ALS and Proposed ALS-PSO for runtime and elapsed time.

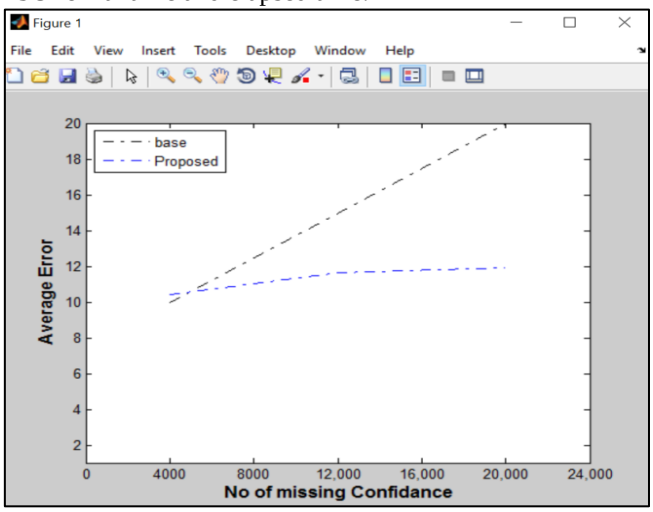

Fig. 1: Comparison between Base and proposed method for average error vs. no of missing confidence

As clearly figure shows the proposed method gain less error and missing confidence as compare to base technique.

A. Fitness Function

The fitness function mentioned below

$$fit(i)=\alpha_1[\frac{Sup(AUC)}{Sup(A)}].[\frac{Sup(AUC)}{Sup(C)}].[1-]+\alpha_2\frac{NumberField(i)}{MaxField}$$

Since the mining association rule is a task that extracts some hidden information, it must discover those rules that have a comparatively less occurrence in the entire database which are more interesting for the users; discovering such rules is more difficult. For classification rules it can be defined by information gain theoretic measures. But it is not efficient for evaluating the association rules.

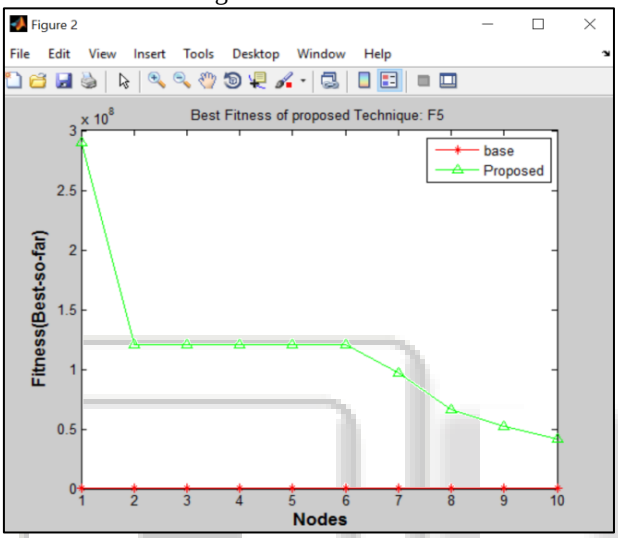

Fig. 2: Comparison between Base and proposed method for best fitness

In above figure our proposed method gain best fitness as compare to Base technique.

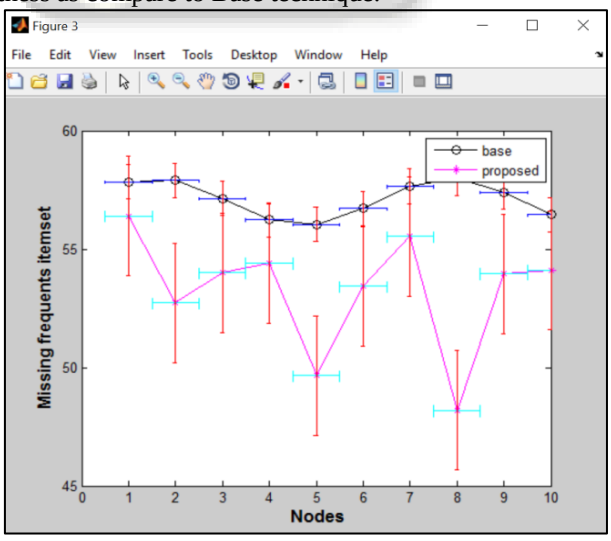

Fig. 3: Comparison between Base and proposed method for missing frequents item set

Also in figure 3 the missing frequent item set also lesser than the base technique.

VI. CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of solving coupled matrix and tensor factorizations when we have squared Euclidean distance as the loss function and introduce pso and low rank approximation, which resolves for all distance factors in all data sets instantaneously. We have also protracted our algorithm to data with missing entrances by addressing the case where we have an incomplete higher-order tensor coupled with matrices. The Apriori-PSO and SVD algorithm can simply be removed to multiple incomplete data sets.

The goal of the future research is to apply this algorithm for real large databases to see the effect of this approach. From the testing results it can be concluded that, for a small number of items random generation of population can give good results. For large number of items, hybrid PSO which can merge genetic algorithm for population generations could give more interesting results which would be the focusing point for further studies.

REFERENCES

[1] LeeSaela, InahJeonb, UKangb, "Scalable Tensor Mining"Big Data Research 2 (2015) 82–86.
[2] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decompositionPsychometrika, 35:283–319, 1970
[3] Cichocki, R. Zdunek, S. Choi, R. J. Plemmons, and S. Amari. Non-negative tensor factorization using alpha and beta divergences In Proceedings of IEEE InternationalConference on Acoustics, Speech, and SignalProcessing, Honolulu, Hawaii, 2007
[4] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. SIAM J.Matrix Anal. Appl., 21(4):1253–1278, 2000.
[5] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "Exploratory" multi-modal factor analysis. UCLA Working Papersin Phonetics, 16:1–84, 1970.
[6] R. A. Harshman. Determination and proof of minimum uniqueness conditions for PARAFAC1UCLAWorking Papers in Phonetics, 22:111–117, 1972.
[7] T. Hazan, S. Polak, and A. Shashua. Sparse image coding using a 3D non-negative tensor factorization. In Proceedings of International Conference on ComputerVision, Beijing, China, 2005
[8] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. Journal of Machine LearningResearch, 5:1457–1469, 2004.
[9] Andersson and R. Bro, The N-way toolbox for MATLAB, Chemometr. Intell. Lab., 52 (2000), pp. 1–4. See also http://www.models. kvl.dk/source/nwaytoolbox/.
[10] R. Bro and H. A. L. Kiers, A new efficient method for determining the number of components in PARAFAC models, Journal of Chemometrics, 17 (2003), pp. 274{286.

