Multi-Level Cluster Indicator Decompositions of Matrices and Tensors

Dijun Luo, Chris Ding, Heng Huang
The University of Texas at Arlington, Arlington, Texas, USA
dijun.luo@gmail.com, chqding@uta.edu, heng@uta.edu

Abstract

A main challenging problem for many machine learning and data mining applications is that the amount of data and features are very large, so that low-rank approximations of original data are often required for efficient computation. We propose new multi-level clustering based low-rank matrix approximations which are comparable and even more compact than Singular Value Decomposition (SVD). We utilize the cluster indicators of data clustering results to form the subspaces, hence our decomposition results are more interpretable. We further generalize our clustering based matrix decompositions to tensor decompositions that are useful in high-order data analysis. We also provide an upper bound for the approximation error of our tensor decomposition algorithm. In all experimental results, our methods significantly outperform traditional decomposition methods such as SVD and high-order SVD.

Introduction

Matrix/tensor decomposition approaches serve as both data compression and unsupervised learning techniques. They have successfully applied in broad applications in artificial intelligence/machine learning domains, including document analysis (Deerwester et al. 1990), bioinformatics (Homayouni et al. 2005), computer vision (Lathauwer, Moor, and Vandewalle 2000; Ding, Huang, and Luo 2008; Ye 2004), inferencing under uncertainty (Wood and Griffiths 2006) and approximate reasoning (Smets 2002) etc. Many other applications were reviewed by Acar and Yener (2008), and Kolda and Bader (2008).

In matrix applications, Singular Value Decomposition (SVD) is the best known and most widely used one, because it provides the best low rank approximation. And in higher order tensors, multi-linear analysis approaches are developed by investigating the projection among multiple factor spaces, *e.g.* High-Order SVD (HOSVD) (Lathauwer, Moor, and Vandewalle 2000; Vasilescu and Terzopoulos 2002) which are popularly used in data mining areas.

As a standard unsupervised learning approach, however, traditional matrix and tensor decomposition approaches often generates results which are not interpretable and further analysis processes are needed. In this paper, one of our major contributions is to propose a decomposition approach which directly outputs interpretable results using clustering based low-rank matrix/tensor approximations. This decomposition uses cluster indicators which has a nice property that they can be compacted into a single vector. We use the indicator space as the subspace bases and project the data onto the spaces. The immediate advantage of this approach is that it costs much less storage. Another advantage is that the matrix/tensor reconstruction process becomes extremely efficient, comparing with tradition decompositions.

We further introduce the multi-scale versions of our method which generate much lower approximation error. Our Cluster Indicator Decomposition (CID) and Multi-Level Cluster Indicator Decomposition (MLCID) are compact (comparable and even more compact than SVD. Figure 1 shows the examples to visually compare the original images, SVD reconstruction results, and MLCID reconstruction results. The results of our method are obviously better than the results of SVD. The details of experimental set up can be found in Experiments section. We also generalize our MLCID methods to tensor decompositions that performs clustering on each dimension and uses clustering indicators to consist of subspaces of tensor decomposition.

Another contribution in the paper is the theoretical analysis of the proposed approach, which provides a tight upper bound of the decompositions. Empirical results show that our MLCID and tensor MLCID decompositions outperform state-of-the-art methods such as SVD and HOSVD decompositions in data sets with clustering structure.

Related Work

We first introduce two traditional matrix and tensor decomposition approaches SVD and HOSVD. The uninterpretable deficiency of decomposition results of these methods intuits us to propose the new matrix and tensor decomposition methods.

SVD

SVD (Singular Value Decomposition) factorizes a given matrix in the following form,

$$X = \sum_{i=1}^{L} \sigma_i u_i v_i^T, \tag{1}$$

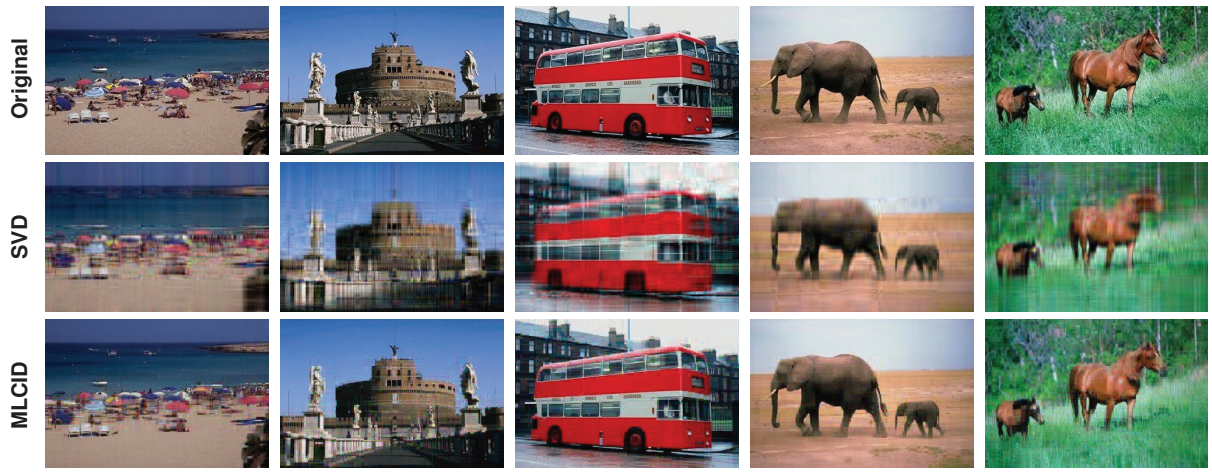


Figure 1: Color image reconstruction results of SVD and MLCID on four randomly selected images from WANG's image database (Li and Wang 2003).

where X is the input matrix and L is the rank of the decomposition. In SVD, u_i are orthonormal: $u_i^T u_j = 1$ if $i = j$, 0 otherwise, as well as v_i.

It is well known that the solution of the following optimization problem is given by SVD:

$$\min_{U,V} J_{SVD} = ||X - UV^T||^2$$
$$s.t.\ U^T U = I, V^T V = I, \quad (2)$$

which means SVD is the best rank-L factorization of any matrix. This is true for any matrix including random matrix.

What would happen in structured data? We show that in well structured data, we are able to factorize a matrix into several low-ranked matrices, some of which become very sparse (we need much less storage to handle sparse matrices). This phenomenon offers a new view of matrix decomposition: SVD might not be the Bible, we might beat SVD.

HOSVD

Some of the data are organized in high order tensor form such as video data. High Order SVD (HOSVD) was presented (Lathauwer, Moor, and Vandewalle 2000) to factorize such data in high order tensors.

Given a 3D tensor $X = \{X_1, \cdots, X_{n_3}\}$ where each X_i is a 2D matrix of size $n_1 \times n_2$, the standard HOSVD factorizations is to solve:

$$\min_{U,V,W,S} J_{HOSVD} = ||X - U \otimes_1 V \otimes_2 W \otimes_3 S||^2$$
$$s.t.\ U^T U = I, V^T V = I,\ W^T W = I \quad (3)$$

where U, V, W are 2D matrices (the sizes are $n_1 \times k_1, n_2 \times k_2$, and $n_3 \times k_3$, respectively), S is a 3D tensor: $S \in \Re^{k_1 \times k_2 \times k_3}$, and $\otimes_n$ is the n-th mode Tucker production. More explicitly,

$$(U \otimes_1 V \otimes_2 W \otimes_3 S)_{ijk} = \sum_{i'j'k'} U_{ii'} V_{jj'} W_{kk'} S_{i'j'k'}.$$

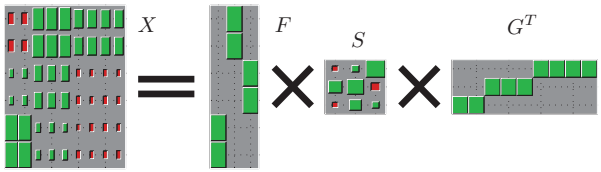


Figure 2: Demonstration of CID decomposition.

One of the difficulties of HOSVD is that the resulting subspaces U, V, and W are not interpretable. The columns are mixed sign and no immediate conclusions can be made. In practical applications, more processes are required in any further analysis, see (Ding, Huang, and Luo 2008) for example. This is similar to SVD.

Multi-Level Cluster Indicator Decomposition

We first introduce a Cluster Indicator Decomposition (CID) (Luo, Ding, and Heng 2011) for two dimensional matrix. After that, we describe the recursive CID and multi-level CID decompositions.

Cluster Indicator Decomposition

For any input matrix X, we seek the decomposition

$$X \cong FSG^T. \quad (4)$$

Here elements of X, S could have mixed signs. We restrict F, G to strictly nonnegative.

The factors F, S, G are obtained from the optimization

$$\min_{F \geq 0, G \geq 0, S} \|X - FSG^T\|^2,$$
$$s.t.\ F^T F = I,\ G^T G = I, \quad (5)$$

where the size of matrices are: $F_{m \times k_1}, S_{k_1 \times k_2}, G_{n \times k_2}$.

It has been shown (Ding et al. 2006) that the above model provides a model for simultaneous clustering of the rows and

columns of X; F, G play the roles of the cluster membership indicators: F for row clustering and G for column clustering.

In this paper, we propose to use the clustering model (based on Eq. (5)) to provide a generic low rank decomposition of matrices, in a similar way as SVD does. As one of the major contributions in this paper, we explicitly require that F, G be exact cluster indicators, *i.e.* each row of F and G has only one non-zero element (which is "1"). This requirement leads to the following advantages:
(1) The vectors F, G have clear meaning: they are exact cluster indicators.
(2) F and G is compact.
(3) Reconstruction of X becomes efficient.

We call this decomposition as Cluster Indicator Decomposition (CID), since we use the clustering indicators in data compression. We give a toy example to illustrate the main concept of CID decompositions. In Figure 2, there is clear cluster structure in input data X. After applying CID decomposition, F and G capture the clustering indictors. One can prove that the solutions F and G in Eq.(5) satisfy $F^TF = I,\ G^TG = I$. Instead of storing them, we use $\tilde{F}$ and $\tilde{G}$:

$$\tilde{F} = Discretize(F)$$
$$\tilde{G} = Discretize(G) \qquad (6)$$

$Discretize(F)$: for each row of F, set the largest element to 1 and the rest to zero[1]. Similar to $Discretize(G)$. F and G can be easily recovered by $\tilde{F}$ and $\tilde{G}$. The algorithm of CID will be described in the end of this section.

Storage Discussion. The key idea behind the CID is the following. Due to the nonnegativity and orthogonality, each row of F has only one 1. Therefore the m-by-k_1 matrix F can be stored in a single m-vector. Similarly, the n-by-k_2 matrix G can be stored in a single n-vector.

The number of row clusters k_1 is usually very small (typically ≤ 10). We pack this integer vector into $b_1 = \lceil \log_2 k_1 \rceil$ bits in the m-vector for F.

Similarly, the $n \times k_2$ matrix G is represented by a n-vector of $b_2 = \lceil \log_2 k_2 \rceil$ bits. Thus, (F, G) requires $mb_1 + nb_2$ bits storage, which is much less than $64(m+n)$ bits storage for a pair of singular vectors (u_l, v_l) using typical machine precision.

In addition, CID also needs the storage for the $k_1 \times k_2$ matrix S. This quantity is negligible because we are interested in the case (m, n) are far larger than (k_1, k_2).

For example, let $k_1 = k_2 = 8$. CID needs $3(m+n)$ bits per iteration [we assume $\min(m, n) \gg \max(k_1, k_2)$]. Thus, CID with $L = 21$ occupies the same storage as a single SVD term.

We can easily show

$$J_{CID} = \sum_{p=1}^{k_1} \sum_{q=1}^{k_2} \left(\sum_{i \in R_p} \sum_{j \in C_q} (X_{ij} - S_{pq})^2 \right) \qquad (7)$$

[1]One can also apply the same strategy to SVD. However, since SVD does not explicitly lead to indicator (for U and V in $X \approx USV$), it generates much higher representation errors.

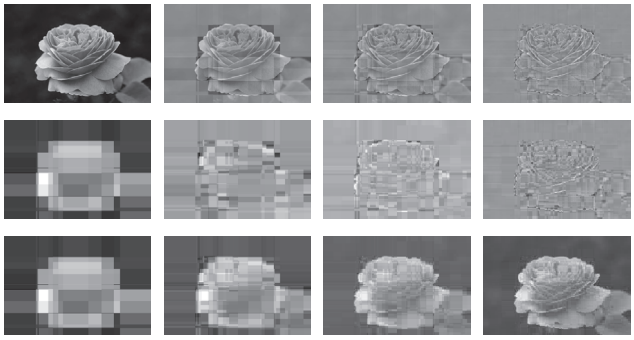


Figure 3: Image reconstruction results of MLCID on one random selected image from WANG's image database. SVD reconstructed image is also plotted at the left bottom corner.

where R_p is the p-th row cluster and C_q is the q-th column cluster. This model is essentially a block clustering model and S_{pq} is the mean value of the block.

Reconstruction Complexity In many kernel machines (e.g. Support Vector Machines) we have to reconstruct the kernel frequently. In this case, one of the bottleneck problems is to fetch kernel values in an efficient way. Here we show that CID has significant advantage in efficiency. We approximate X by the following,

$$X \approx (FSG^T)_{ij} = \sum_{pq} S_{pq} F_{ip} G_{jq} = S_{\mathbf{f}_i \mathbf{g}_j}, \qquad (8)$$

where $\mathbf{f}, \mathbf{g}$ are single-column indicators: $\mathbf{f}_i = p$ if $F_{ip} = 1$, $i = 1, 2, ..., m$ and $\mathbf{g}_j = q$ if $G_{jq} = 1$, $j = 1, 2, ..., n$. This indicates that the computational complexity of every entry X_{ij} is just two steps of indexing (addressing): as if all entries are pre-computed. This is useful when solving large scale problems in which pre-computed kernels could not be stored in fast memory.

Recursive CID Decomposition

Recursive CID performs repeated orthogonal-NMF clustering model on the input data X. The complete recursive CID is

$$X = \sum_{l=1}^{L} F_l S_l G_l^T. \qquad (9)$$

Recursive CID is computed using deflection. We first compute orthogonal NMF clustering of X to obtain F_1, S_1, G_1 using the same algorithm in the end of this section. We set $X_1 = X - F_1 S_1 G_1^T$ and apply the algorithm on X_1 to obtain F_2, S_2, G_2. This is repeated L times.

We note that SVD of X can be written as $X = \sum_{l=1}^{L} u_l \sigma_l v_l^T$. where F_l has the same storage as u_l and G_l has the same storage as v_l. Thus, we can compare these reconstruction errors of these two decompositions at same L.

Multi-level CID Decomposition

In recursive CID, we iteratively do matrix decomposition based on NMF clustering results. From the second iteration, the clustering algorithm is always performed on the

residue between data matrix X_l (residue from previous iteration) and decomposition result $F_{l+1}S_{l+1}G_{l+1}^T$. We notice the clustering structures in residues are less clear than the structure in original data matrix. In order to improve the performance of matrix decompositions, we propose to use multi-level CID to efficiently seek clustering structures in residues.

Although the clustering structures are weak in residues globally, they still often clearly exist in local regions of residues. In MLCID, we first apply CID on X to obtain F_1, S_1, G_1.

In second iteration, we split the residue matrix $\Delta X = X - F_1 S_1 G_1^T$ into 2×2 equal area blocks

$$\begin{pmatrix} \Delta X_{11} & \Delta X_{12} \\ \Delta X_{13} & \Delta X_{14} \end{pmatrix},$$

and apply CID to each sub-matrix.

In iteration l, we split the current $2^{2*(l-1)}$ blocks to 2^{2*l} smaller equal area blocks and apply CID to each block. The level l MLCID decomposition is finished in l-th iteration. The same step is repeated till the iteration number l reach the given level number L.

If the number of row/column (m or n) is not even, we just split them into $(m+1)/2$ and $(m-1)/2$, or $(n+1)/2$ and $(n-1)/2$. Meantime, the total iteration number L is given by user. The storage of MLCID is $k_1k_2(2^{(2L)}-1)/3 + (2^{l+1}-1)(n\log_2 k_1 + m\log_2 k_2)/64$, where the first term is for the S factors, and the second term is for F and G indicators.

Figure 3 shows one example of MLCID decomposition. MLCID has been performed on one random selected image from WANG's image database (Li and Wang 2003). $L = 4$ is used. Figures in the first row are input image of each level (residues of previous level). Figures in the second row are CID approximation of current level only. Figures in the third row show the MLCID reconstruction images.

Figure 1 illustrates the image reconstruction comparison between MLCID and SVD. We use four color images from WANG's image corpus (Li and Wang 2003). Please see the detailed experimental setup in the experimental results section. The first row of the figure lists the original images of WANG's image dataset. The second row includes the reconstruction results of SVD, and the bottom row shows the results of our MLCID decomposition approach. In all four images, MLCID gives out much more clear reconstructed images compared to SVD.

CID Algorithm

The main algorithm of CID is

(A0) Initialize F, G using (S0)

(A1) Update S, F, G until convergence using (S1)

(A2) Discretize F, G using (S2)

(A3) Update S until convergence. using (S1)

(A4) Pack F as a vector. Pack G as a vector.

We compute the Semi-NMF factorization via an iterative updating algorithm that alternatively updates F and G:

(S0) Initialize G. Do K-means clustering to cluster columns of X into k_2 clusters. This gives cluster indicator G: $G_{ik} = 1$ if $\mathbf{x}_i$ belongs to cluster k. Otherwise, $G_{ik} = 0$. Let $G_0 = G + 0.2$. Normalize each column of G_0 to 1 using L_2-norm (thus $G_0^T G_0 \cong I$). Initialize F in same way by clustering rows of X into k_1 clusters.

(S1) Update S, F, G using the rule

Repeat (S1a) - (S1c) until convergence:

(S1a) Compute S (while fixing F, G) using

$$S = (F^T F)^{-1} F^T X G (G^T G)^{-1}. \tag{10}$$

Note that computing $(F^T F)^{-1}, (G^T G)^{-1}$ are easy tasks because $G^T G$ and $F^T F$ are $k_1 \times k_1$ and $k_2 \times k_2$ positive semidefinite matrices. The inversion of these small matrices is trivial.

(S1b) Update G (while fixing $F \Leftarrow FS$) using

$$G_{ik} \leftarrow G_{ik} \sqrt{\frac{(X^T F)^+_{ik}}{(X^T F)^-_{ik} + [G(F^T F)]_{ik}}}, \tag{11}$$

where positive and negative parts of matrix A are

$$A^+_{ik} = (|A_{ik}| + A_{ik})/2,\ A^-_{ik} = (|A_{ik}| - A_{ik})/2. \tag{12}$$

(S1c) Update F (while fixing $G \Leftarrow GS$) using

$$F_{ik} \leftarrow F_{ik} \sqrt{\frac{(XG)^+_{ik}}{(XG)^-_{ik} + [F(G^T G)]_{ik}}}, \tag{13}$$

(S2) Discretize F: for each row of F, set the largest element to 1 and the rest to zero. Discretize G similarly. Note this sparsity pattern remains unchanged during the updating of (S1b) and (S1c).

Tensor Clustering Indicator Decomposition

Our CID, recursive CID, and MLCID decompositions can be generalized to high dimensional tensor decompositions. Similar to other multilinear analysis methods, tensor CID still searches low ranked matrixes and a core tensor to approximate high order tensors.

Tensor CID Decomposition

Given a 3D tensor $X = \{X_1, \cdots, X_{n_3}\}$ where each X_i is a 2D matrix of size $n_1 \times n_2$, the tensor CID is to solve:

$$\begin{aligned} &\min_{U,V,W,S}\ J_1 = ||X - U \otimes_1 V \otimes_2 W \otimes_3 S||^2 \\ &s.t.\ \ U^T U = I, V^T V = I,\ W^T W = I \\ &\qquad U \geq 0, V \geq 0, W \geq 0 \end{aligned} \tag{14}$$

where U, V, W are 2D matrices (consisting of clustering indicators) and S is a 3D tensor: $S \in \Re^{k_1 \times k_2 \times k_3}$.

Tensor CID algorithm is:

(B0) Do K-means clustering on each direction of tensor X to get clustering indicator matrices U, V, and W

(B1) Discretize U: for each row of U, set the largest element to 1 and the rest to zero. Normalize each column of U to 1 using L_2-norm (thus $U^T U = I$). Similar to discretize V and W.

(B2) Calculate $S = U^T \otimes_1 V^T \otimes_2 W^T \otimes_3 X$

The tensor CID decomposition result is

$$X = U \otimes_1 V \otimes_2 W \otimes_3 S \tag{15}$$

Tensor Recursive CID Decomposition

Tensor recursive CID algorithm is:

(C0) Given total iteration number L, starting from $l = 1$, and $X_1 = X$

(C1) Do tensor CID on tensor X_l to get U_l, V_l, W_l, and S_l

(C2) Calculate residue $X_{l+1} = X_l - U_l \otimes_1 V_l \otimes_2 W_l \otimes_3 S_l$

(C3) If the iteration number is beyond given number L, then stop. Otherwise, let $l = l + 1$ and go to (C1).

The complete tensor recursive CID decomposition is

$$X = \sum_{l=1}^{L} U_l \otimes_1 V_l \otimes_2 W_l \otimes_3 S_l \tag{16}$$

Tensor MLCID Decomposition

Tensor MLCID algorithm is:

(D0) Given multi-level number L, starting from level $l = 1$, set up tensor set $Y_1 = \{X\}$, assuming the number of tensors in tensor set Y_l is N_l

(D1) Do tensor CID on each tensor in Y_l to get U_{li}, V_{li}, W_{li}, and S_{li}, $i = 1, \cdots, N_l$

(D2) If the level number is beyond given number L, then stop. Otherwise, uniformly split each tensor in Y_l to $2 \times 2 \times 2$ sub-tensors and create a new tensor set Y_{l+1} to include all these sub-tensors. Let $l = l + 1$ and go to (D1).

Error Analysis for Tensor CID

We provide a theoretical analysis of our proposed approach. Since CID is a special case of tensor CID, we only show the analysis of tensor CID here. Given a 3D tensor: $X = \{X_{ijk}\}_{i=1\ j=1\ k=1}^{n_1\ n_2\ n_3}$, we denote

$$I_{slab} = \{X_{i,j,k} | 1 \le j \le n_2, 1 \le k \le n_3\},$$
$$J_{slab} = \{X_{i,j,k} | 1 \le i \le n_1, 1 \le k \le n_3\},$$
$$K_{slab} = \{X_{i,j,k} | 1 \le i \le n_1, 1 \le j \le n_2\}.$$

Let J_{TCID} be the reconstruction error of the tensor CID decomposition. Let $J_{Kmeans}^{I_{slab}}$ be the error in K-means clustering of the I_{slab} of X, $J_{Kmeans}^{J_{slab}}$ be the error in K-means clustering of the J_{slab} of X, and $J_{Kmeans}^{K_{slab}}$ be the error in K-means clustering of the K_{slab} of X. We have the following

Theorem 1 *In tensor CID composition, we have*

$$J_{TCID} \le J_{Kmeans}^{I_{slab}} + J_{Kmeans}^{J_{slab}} + J_{Kmeans}^{K_{slab}}. \tag{17}$$

Proof.

$$\begin{aligned} J_{TCID} &= ||X - U \otimes_1 V \otimes_2 W \otimes_3 S|| \\ &\le ||X - U \otimes_1 S'|| + ||U \otimes_1 S' - U \otimes_1 V \otimes_2 S''|| \\ &\quad + ||U \otimes_1 V \otimes_2 S'' - U \otimes_1 V \otimes_2 W \otimes_3 S|| \\ &= ||X - U \otimes_1 S'|| + ||S' - V \otimes_2 S''|| \\ &\quad + ||S'' - W \otimes_3 S|| \\ &\le J_{Kmeans}^{I_{slab}} + J_{Kmeans}^{J_{slab}} + J_{Kmeans}^{K_{slab}}. \end{aligned} \tag{18}$$

■

For tensor MLCID, Theorem 1 is held on each level decomposition.

Experimental Evaluations

We present empirical results to show the efficiency of our decomposition algorithms. In our experiments, we always compare the performance of different methods over the same storage, and the approximation error is computed as $\epsilon = \|X - \hat{X}\|^2 / \|X\|^2$, where $\hat{X}$ is the reconstruction of X.

CID Decomposition for Images

We first perform decompositions on images to visualize the approximation capability comparing to SVD. Here we randomly select four color images from WANG's image corpus (Li and Wang 2003). In this experiment, we use the original size of the image (384×256) and divided the color images into three channel images (red, green, and blue channels). Decomposition approaches are applied to all three channels independently. After we get the reconstructed images, we combine the R,G,and B channels together to form a color images, which are shown in Figure 1. The first row of the figure lists the original images of WANG's image data set. The second row includes the reconstruction results of SVD, and the bottom row shows the results of our MLCID decomposition approach.

For MLCID, we choose $k_1 = k_2 = 8$ and $L = 4$ for all the images. In this case, the total storage for MLCID is $5440 + 690 = 6130$. For SVD, we choose $k = 9$ and the corresponding storage is $k(m + n) + k = 5769$. In all four images of Figure 1, MLCID gives out much better reconstructed images compared to SVD. These results indicate that MLCID is more compact than SVD.

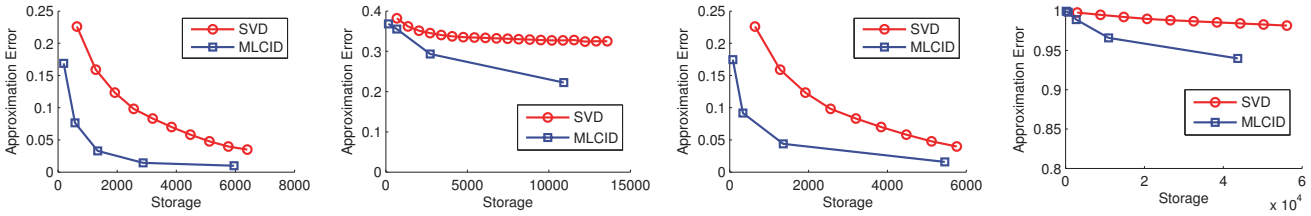
Systematic Results for MLCID

Here we investigate the reconstruction error of MLCID decomposition using four data sets: two data sets from UCI (ECOLI, YEAST), one image data set from WANG's image data sets, and one from 20 News Group. For 20 News Group, we used the first 2000 documents and use F-score to select the top 1000 related words to form a 2000×1000 matrix. The matrix size of the data can be found in the first part of Table 1.

Table 1: Data sets statistics used in our experiment.

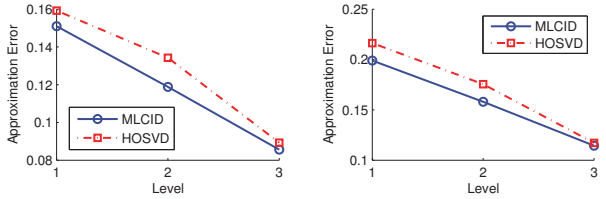
Data set	n_1	n_2	n_3
20NewsGroup	2000	1000	
ECOLI	336	343	
WANG	256	384	
YEAST	1484	1470	
WANG3D	128	192	100
Wimbledon	468	850	100

The comparison results can be found in Figure4. In all data sets, our MLCID method provides better matrices approximations with lower reconstruction errors compared to SVD. Notice that in data set Yeast, SVD is inefficient when the size of storage is small. But compared to SVD (about 0.98), MLCID generates much lower reconstruction errors (about 0.74) .



(a) 20News (b) Ecoli (c) WANG (d) Yeast

Figure 4: Approximation error comparison of SVD and MLCID under the same storage for 20NewsGroup, ECOLI, WANG, and YEAST data.



(a) WANG (b) Wimbledon

Figure 5: Approximation error comparison for tensor MLCID and HOSVD on tensor constructed in WANG's and Wimbledon data sets.

Performance Evaluations on Tensor MLCID

We compare the compression capability of MLCID with HOSVD using two data sets. The first one (WANG3D) is from WANG's image database. We randomly pick 100 images with resolution 256×384, resize them into 128×192, and transfer them into gray level images. The final tensor is $128 \times 192 \times 100$. The other data set is extracted from video of a final match in Wimbledon 2009 tennis championship. We resize the frames into 468×850 and pick up 100 frames. The final tensor size of the data is $468 \times 850 \times 100$. We set $K_1 = K_2 = K_3 = 4$ and $L = [1, 2, 3]$ for tensor MLCID decomposition. And compare the tensor data reconstruction errors under the same storage. Results are plotted in Figures 5. Our tensor CID has lower reconstruction errors on tensor data than HOSVD. When the storage is increased, HOSVD will gradually get good reconstruction results. But it will lose the low-rank approximation purpose.

Conclusion

In this paper, we show that data clustering can be used to derive effective low rank matrix decompositions which are both compact and interpretable. We proposed CID, recursive CID, and MLCID matrix decomposition methods. Moreover, we also generalize them to tensor decompositions. The empirical experimental studies show our methods outperform the traditional SVD method. Our approaches open a new application area for data clustering and efficiently solve the data low-rank approximation problem existing in many large-scale machine learning and data mining applications.

Acknowledgments This research was supported by NSF-CCF 0830780, NSF-CCF 0917274, NSF-DMS 0915228.

References

Acar, E., and Yener, B. 2008. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*.

Deerwester, S.; Dumais, S.; Landauer, T.; and Harshman, G. F. R. 1990. Indexing by latent semantic analysis. *J. Amer. Soc. Info. Sci* 41:391–407.

Ding, C. H. Q.; Li, T.; Peng, W.; and Park, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, 126–135. ACM.

Ding, C. H. Q.; Huang, H.; and Luo, D. 2008. Tensor reduction error analysis - applications to video compression and classification.

Homayouni, R.; Heinrich, K.; Wei, L.; and Berry, M. W. 2005. Gene clustering by latent semantic indexing of MEDLINE abstracts. *Bioinformatics* 21(1):104–115.

Kolda, T. G., and Bader, B. W. 2008. Tensor decompositions and applications. *SIAM Review*.

Lathauwer, L. D.; Moor, B. D.; and Vandewalle, J. 2000. On the best rank-1 and rank-(r1, r2, . . . , rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.* 21:1324–1342.

Li, J., and Wang, J. Z. 2003. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(9):1075–1088.

Luo, D.; Ding, C.; and Heng, H. 2011. Cluster indicator decomposition for efficient matrix factorization. *Proc. Int'l Joint Conf on Artificial Intelligence*.

Smets, P. 2002. The application of the matrix calculus to belief functions. *Int. J. Approx. Reasoning* 31(1-2):1–30.

Vasilescu, M., and Terzopoulos, D. 2002. Multilinear analysis of image ensembles: Tensorfaces. *European Conf. on Computer Vision* 447–460.

Wood, F., and Griffiths, T. L. 2006. Particle filtering for nonparametric bayesian matrix factorization. 1513–1520.

Ye, J. 2004. Generalized low rank approximations of matrices. *International Conference on Machine Learning*.