Cluster Indicator Decomposition for Efficient Matrix Factorization

Dijun Luo, Chris Ding, Heng Huang
Computer Science and Engineering Department, University of Texas at Arlington, USA
dijun.luo@gmail.com; chqding@uta.edu; heng@uta.edu

Abstract

We propose a new clustering based low-rank matrix approximation method, Cluster Indicator Decomposition (CID), which yields more accurate low-rank approximations than previous commonly used singular value decomposition and other Nyström style decompositions. Our model utilizes the intrinsic structures of data and theoretically be more compact and accurate than the traditional low rank approximation approaches. The reconstruction in CID is extremely fast leading to a desirable advantage of our method in large-scale kernel machines (like Support Vector Machines) in which the reconstruction of the kernels needs to be frequently computed. Experimental results indicate that our approach compress images much more efficiently than other factorization based methods. We show that combining our method with Support Vector Machines obtains more accurate approximation and more accurate prediction while consuming much less computation resources.

1 Introduction

Many machine learning applications require processing large amounts of high dimensional data. In these applications, the data are represented as a $m \times n$ matrix $\mathbf{X}$, *e.g.* images, videos, kernel matrices, spectral graph. As data size and the amount of redundancy increase fast with dimensionality m and n, it is desirable to obtain compact and concise representations of $\mathbf{X}$ to make data analysis and interpretation easier, *e.g.*, a low-rank approximation of original data.

Low-rank approximation of matrices have been widely used in broad artificial intelligence applications such as image denoising and compression [Zhang *et al.*, 2009], face recognition [Turk and Pentland, 1991], motion scene reconstruction [Han and Kanade, 2001], scene appearance approximation [Garg *et al.*, 2009], and part of objects representation [Lee and Seung, 1999] *etc*. Among them, kernel matrix approximation is one of the most important applications. Kernel methods play a central role to successfully model computer vision data with highly complex, nonlinear structures, *e.g.* support vector machine, kernel Fisher discriminant analysis, and kernel principal component analysis. But the computational complexities of large kernel matrices in terms of both space (quadratic) and time (usually cubic) are quite challenge in practical applications. Low-rank approximations of the kernel matrix can effectively tackle large-scale datasets with no significant decrease in performance [Williams and Seeger, 2000; Zhang *et al.*, 2008; Fowlkes *et al.*, 2004; Talwalkar *et al.*, 2008].

There are a very large number of different low rank approximation methods [Bach and Jordan, 2005; Frieze *et al.*, 2004], among them Singular Value Decomposition (SVD) is the best known and most widely used one, because it provides the best low rank approximation so far and can be readily computed.

Another popular low-rank approximation algorithm is Nyström decompositions, because it is easy to be computed and also the vectors involved in factorization are "interpretable". Nyström style decompositions have been widely used to speed up the computation of large kernel matrix [Williams and Seeger, 2000; Zhang *et al.*, 2008; Drineas and Kannan, 2003; Fowlkes *et al.*, 2004; Srebro and Jaakkola, 2003].

In this paper, we propose a new clustering based low-rank matrix approximation. This decomposition uses cluster indicators which have a nice property that they can be stored into a single vector, while provide the approximation capability of rank-K approximation. This Cluster Indicator Decomposition (CID) is thus compact (comparable and even more compact than SVD). Meantime, it is also interpretable because the basis vectors are cluster indicators.

The clustering model here also naturally suggests a Relaxed Cluster Indicator Decomposition where the nonzero elements of cluster indicators are not forced to be "1". Our new low-rank matrix approximation methods have close connection to Nyström style decompositions and several other closely related variants. But our methods have much better approximation accuracy than related approaches. Using four standard image datasets, we evaluate the performance of our methods by image compression and kernel matrix approximation applications in term of reconstruction errors and classification accuracy. We provide a new insight to low-rank matrix approximation and our methods can be widely applied into many computer vision applications.

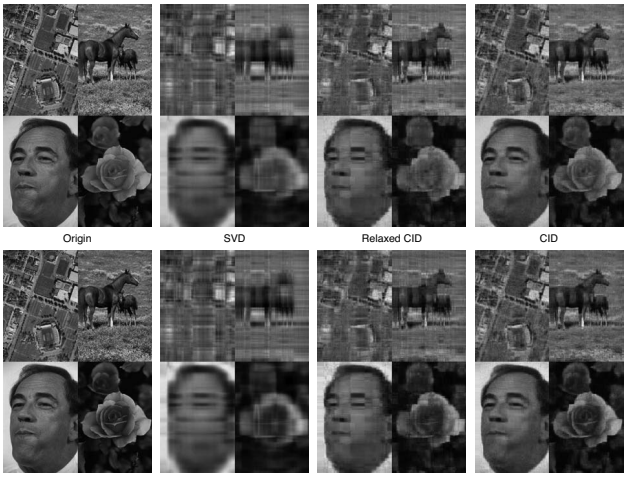


Figure 1: Image reconstruction using SVD, CID, and Relaxed CID on color images (top) and gray level images (bottom). Four images are from diverse data source (see §4.1). For SVD, $k = 4$; parameters for CID and Relaxed CID are set such that all methods use the same storage. The reconstruction errors of SVD, Relaxed CID, CID are 2.98%, 2.22%, 1.51% for color images, and 6.27%, 4.43%, 2.95% for gray level images. CID has the best results and Relaxed CID is also better than SVD.

2 Cluster Indicator Decomposition (CID)

For any rectangular (or non-symmetric square) input matrix $\mathbf{X} \in \Re^{m\times n}$, we seek the CID decomposition of $\mathbf{X}$ as

$$\mathbf{X} \cong \mathbf{F}\mathbf{S}\mathbf{G}^T, \;\; \mathbf{F} = \{0,1\}^{m\times k_1}, \tag{1}$$

$$\mathbf{S} \in \Re^{k_1\times k_2}, \; \mathbf{G} = \{0,1\}^{n\times k_2}. \tag{2}$$

In addition, each row of $\mathbf{F}$ has exactly one nonzero element "1". Thus $\mathbf{F} = (\mathbf{f}_1, \cdots, \mathbf{f}_{k_1})$ is the cluster indicator matrix for row clustering of $\mathbf{X}$, *i.e.*, the non-zeros of $\mathbf{f}_p$ indicate rows of $\mathbf{X}$ which form a row cluster R_p. Similarly, $\mathbf{G} = (\mathbf{g}_1, \cdots, \mathbf{g}_{k_2})$ is the cluster indicator matrix for column clustering of $\mathbf{X}$, where the non-zeros of $\mathbf{g}_q$ indicate columns of $\mathbf{X}$ forming a column cluster C_q. The factors $\mathbf{F}, \mathbf{S}, \mathbf{G}$ are obtained by

$$\min_{\mathbf{F},\mathbf{G},\mathbf{S}} J_0 = \|\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^T\|^2, \tag{3}$$

which can be equivalently written as

$$\min \; J_0 = \sum_{i=1}^{n}\sum_{j=1}^{m} \min_{1\le p\le k_1} \min_{1\le q\le k_2} (\mathbf{X}_{ij} - \mathbf{S}_{pq})^2. \tag{4}$$

This can be viewed as an extension of the standard "one-way" K-means clustering:

$$\min \; J^{\text{columns}}_{\text{Kmeans}}(\mathbf{X}) = \sum_{q=1}^{k}\sum_{i\in C_q} \|\mathbf{x}_i - \mu_q\|^2 = \sum_{i=1}^{n} \min_{1\le q\le k} \|\mathbf{x}_i - \mu_q\|^2 \tag{5}$$

where μ_q is the centroid of cluster C_q. The simultaneous clustering of rows and columns of $\mathbf{X}$ can be viewed as *block* clustering and $\mathbf{S}_{pq}$ is the "mean" of the block cluster.

Storage of CID

CID is very compact. The indicator matrix $\mathbf{F}$ can be stored in an m-vector $\mathbf{f}_{\text{CID}}$ of b_1-bit integers, where $b_1 = \lceil \log_2 k_1 \rceil$ and $\mathbf{f}_{\text{CID}}(j)$ indicate the index of row cluster that j-th row belongs to. Similarly, indicator matrix $\mathbf{G}$ can be stored in an n-vector $\mathbf{g}_{\text{CID}}$ of b_2-bit integers where $b_2 = \lceil \log_2 k_2 \rceil$. $(\mathbf{F}, \mathbf{G})$ requires $mb_1 + nb_2$ bits storage, which is much less than $64(m+n)$ bits storage for a pair of singular vectors $(\mathbf{u}_l, \mathbf{v}_l)$. For this reason, we sometimes call it rank-0 storage. $\mathbf{S}$ is stored as k_1k_2 *real* numbers. Thus the total storage Γ for CID in unit of 64-bit *reals* is

$$\Gamma_{\text{CID}} = m\frac{\lceil \log_2 k_1 \rceil}{64} + n\frac{\lceil \log_2 k_2 \rceil}{64} + k_1k_2 \tag{6}$$

We note that for SVD truncated into K_{SVD} terms, the storage is $\Gamma_{\text{SVD}} = K_{SVD}(m+n)$. A very simple implementation uses a whole 64-bit integer for $\mathbf{F}$ and other 64-bit integer for $\mathbf{G}$. This simple CID use storage less than SVD if

$$k_1k_2 < (K_{\text{SVD}} - 1)(m+n). \tag{7}$$

2.1 Relaxed CID

In CID, each row of $\mathbf{F}$ has exactly one nonzero element which is "1". We now relax this nonzero element to a real number. This improves the matrix approximations, but increase storage slightly. We can also relax $\mathbf{G}$ similarly. We call this decomposition as Relaxed CID.

Storage of Relaxed CID

Because each row of $\mathbf{F}$ has only one nonzero, the k_1 columns of $\mathbf{F} = (\mathbf{f}_1, \cdots, \mathbf{f}_{k_1})$ can be stored in a single m-vector as $\mathbf{f}_{\text{RCID}}$ Cluster labels of each row is stored in $\mathbf{f}_{\text{CID}}$ (see storage of CID). Furthermore, because each element of $\mathbf{f}_{\text{CID}}$ has only $b_1 = \log k_1$ bits (for k_1=32 clusters, $b_1 = 5$), $\mathbf{f}_{\text{CID}}$ can be packed in to the least significant b_1-bits in $\mathbf{f}_{\text{RCID}}$'s mantissa. $\mathbf{G}$ is stored similarly. Over-all, the storage of $\mathbf{F}, \mathbf{G}$ is equivalent to storage of a pair singular vectors of SVD (we may call Relaxed CID as "CID with rank-1 storage").

2.2 Computational algorithm of RCID and CID

Optimization of Eq. (3) with discrete constraints of Eqs. (1,2) is an NP-hard combinatorial optimization. Here we use a continuous approach to first find a relaxed solution.

We do a transformation. Let $\tilde{\mathbf{F}} = \mathbf{F}D_f^{-1/2}$ where $D_f = \text{diag}(m_1, \cdots, m_{k_1})$ and $m_p = |R_p|$ is the size of row-cluster R_p. Clearly, $\tilde{\mathbf{F}}^T\tilde{\mathbf{F}} = I$ where I is an identity matrix of appropriate size. Similarly, let $\tilde{\mathbf{G}} = \mathbf{G}D_g^{-1/2}$ where $D_g = \text{diag}(n_1, \cdots, n_{k_2})$ and $n_q = |C_q|$ is the size of column-cluster C_q. Clearly, $\tilde{\mathbf{G}}^T\tilde{\mathbf{G}} = I$. Let $\tilde{\mathbf{S}} = D_f^{1/2}\mathbf{S}D_g^{1/2}$ be the new central factor.

It is obvious that $\tilde{\mathbf{F}}\tilde{\mathbf{S}}\tilde{\mathbf{G}}^T = \mathbf{F}\mathbf{S}\mathbf{G}^T$. But in terms of these new factors, the constraints become easier to deal with. Now, for notational simplicity, we use $\mathbf{F}$ for $\tilde{\mathbf{F}}$, $\mathbf{G}$ for $\tilde{\mathbf{G}}$, $\mathbf{S}$ for $\tilde{\mathbf{S}}$. These factors are obtained from the optimization

$$\min_{\mathbf{F}\ge 0, \mathbf{G}\ge 0, \mathbf{S}\ge 0} \|\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^T\|^2, \;\; s.t.\; \mathbf{F}^T\mathbf{F} = I, \; \mathbf{G}^T\mathbf{G} = I. \tag{8}$$

Since $\mathbf{F}$ is nonnegative, the orthogonality of $\mathbf{F}$ implies each row of $\mathbf{F}$ has only one nonzero. But this nonzero element is

not restricted to "1", as required by a strict "cluster indicator". For this reason, we call $\mathbf{F}$ "relaxed" cluster indicator. Similarly, the relaxed indicator $\mathbf{G}$ is computed in this optimization. Therefore, this optimization gives solution for Relaxed CID.

Once the relaxed indicators are obtained in Relaxed CID, we set the non-zeros in $\mathbf{F}, \mathbf{G}$ to "1" , fixing them, and update $\mathbf{S}$ to obtain the solution for CID.

Here we outline the algorithm [Ding *et al.*, 2006] to solve the orthogonal tri-factorization Eq.(8). Here the data X is nonnegative. The algorithm is given as below:

(S0) Initialize $\mathbf{G}$. Do K-means clustering on columns of $\mathbf{X}$ into k_2 clusters. This gives cluster indicator $\mathbf{G}$ and let $\mathbf{G}_0 = \mathbf{G} + 0.2$. Normalize each column of $\mathbf{G}_0$ to 1 using L_2-norm (thus $\mathbf{G}_0^T\mathbf{G}_0 \cong I$). Initialize $\mathbf{F}$ in same way by clustering rows of $\mathbf{X}$ into k_1 clusters.

(S1) Repeat until convergence:

$$\mathbf{S}_{kl} \leftarrow \mathbf{S}_{kl}\frac{(\mathbf{F}^T\mathbf{X}\mathbf{G})_{kl}}{(\mathbf{F}^T\mathbf{F}\mathbf{S}\mathbf{G}^T\mathbf{G})_{kl}} \tag{9}$$

$$\mathbf{G}_{jk} \leftarrow \mathbf{G}_{jk}\left[\frac{(\mathbf{X}^T\mathbf{F}\mathbf{S})_{jk}}{(\mathbf{G}\mathbf{G}^T\mathbf{X}^T\mathbf{F}\mathbf{S})_{jk}}\right], \tag{10}$$

$$\mathbf{F}_{ik} \leftarrow \mathbf{F}_{ik}\left[\frac{(\mathbf{X}\mathbf{G}\mathbf{S}^T)_{ik}}{(\mathbf{F}\mathbf{F}^T\mathbf{X}\mathbf{G}\mathbf{S}^T)_{ik}}\right] \tag{11}$$

Notice that the RHS factors (square brackets) of Eqs. (10) and (11) are not square rooted as in [Ding *et al.*, 2006]. Thus algorithm here converges faster than that in [Ding *et al.*, 2006]. The convergence of Eqs. (10) and (11) can be vigorously proved (details are skipped due to space limitation).

Real Life Image Compression

Fig. 1 shows the real images compression results using CID and Relaxed CID. The original image and SVD results are also shown. From the figures, we can see that CID compressed images are much more clear than SVD results. The image details are described in the first paragraph of §4.

2.3 Error bounds for CID

Let J^*_{CID} be the optimal reconstruction error of CID and and J^*_{RCID} be the optimal reconstruction error of relaxed CID. We show there exists an easily computable upper bound:

Theorem 2.1 *In CID and RCID compositions of the input* $\mathbf{X}$*, we have the upper bound*

$$J^*_{RCID} \le J^*_{CID} \le J^{columns}_{Kmeans}(\mathbf{X}) + J^{rows}_{Kmeans}(\mathbf{Y}). \tag{12}$$

where $J^{columns}_{Kmeans}(\mathbf{X}), J^{rows}_{Kmeans}(\mathbf{Y})$ *are obtained as following:*

CID bound calculation. **(1)** Do K-means clustering on the columns of $\mathbf{X}$. Let $J^{\text{columns}}_{\text{Kmeans}}(\mathbf{X})$ be the corresponding objective function value. Let $\overline{\mathbf{Y}}$ be the *centroid data*, *i.e.*, each x_i is replaced by its corresponding cluster centroid:

$$\overline{\mathbf{Y}} = (\overbrace{\mu_1,\cdots,\mu_1}^{n_1}, \overbrace{\mu_2,\cdots,\mu_2}^{n_2}, \cdots, \overbrace{\mu_k,\cdots,\mu_k}^{n_k}) \tag{13}$$

(here we assume, without loss of generality, that columns of $\mathbf{X}$ are ordered such that data points within same cluster are adjacent, and setting $k_1 = k_2 = k$ to simplify the notation). **(2)** Do K-means clustering on rows of the m-by-n matrix $\overline{\mathbf{Y}}$. This is identical to the clustering of the rows of the *condensed centroid data* $\mathbf{Y}$:

$$\mathbf{Y} = (\ n_1\mu_1,\ n_2\mu_2,\ \cdots,\ n_k\mu_k\ ). \tag{14}$$

Clustering on $\mathbf{Y}$ is faster since $\mathbf{Y}$ is size of m-by-k and $k \ll n$. $J^{\text{rows}}_{\text{Kmeans}}(\mathbf{Y})$ is the objective function value of this clustering.

Proof of Theorem 1. The inequality $J^*_{\text{RCID}} \le J^*_{\text{CID}}$ is true because the relaxed indicator $\mathbf{F}, \mathbf{G}$ from "1" improves the approximation. To obtain an upper bound of J_{CID}, it is sufficient to find a feasible solution of the optimization of Eq. (3) or the equivalent block clustering formulation of Eq. (4). Clearly, the solution $(\mathbf{F}_0, \mathbf{G}_0)$ obtained by the CID bound computing algorithm above is a feasible solution of the block clustering. Furthermore, the block clustering objective function value of the solution $(\mathbf{F}_0, \mathbf{G}_0)$ is exactly $J_{\text{CID}}(\mathbf{F}_0, \mathbf{G}_0) = J^{\text{columns}}_{\text{Kmeans}}(\mathbf{X}) + J^{\text{rows}}_{\text{Kmeans}}(\mathbf{Y})$. By definition, the optimal solution should have a lower objective comparing to a feasible solution, *i.e.*, $J^*_{\text{CID}} \le J_{\text{CID}}(\mathbf{F}_0, \mathbf{G}_0)$. ■

2.4 CID for symmetric/kernel matrices

For kernel (and generic symmetric) matrices $\mathbf{W}$, due to symmetry, $\mathbf{F} = \mathbf{G}$. Therefore, the decomposition becomes

$$\mathbf{W} \cong \mathbf{G}\mathbf{S}\mathbf{G}^T, \tag{15}$$

where $\mathbf{G}$ is restricted to be an indicator matrix. As discussed in §2.1,the relaxed indicator $\mathbf{S}$ and factor $\mathbf{G}$ are obtained from the optimization

$$\min_{\mathbf{G}\ge 0, \mathbf{S}\ge 0} \|\mathbf{W} - \mathbf{G}\mathbf{S}\mathbf{G}^T\|^2,\ s.t.\ \mathbf{F}^T\mathbf{F} = I. \tag{16}$$

The algorithm to compute $\mathbf{G}, \mathbf{S}$ is very similar to that of §2.2. We outline the algorithm as following:

(B0) Initialize $\mathbf{G}$ by clustering rows of $\mathbf{W}$.

(B1) Update $\mathbf{S}$: $\mathbf{S}_{kl} = \mathbf{S}_{kl}\frac{(\mathbf{G}^T\mathbf{W}\mathbf{G})_{kl}}{(\mathbf{G}^T\mathbf{G}\mathbf{S}\mathbf{G}^T\mathbf{G})_{kl}}$.

(B2) Update $\mathbf{G}$:

$$\mathbf{G}_{ik} \leftarrow \mathbf{G}_{ik}\Big(\frac{(\mathbf{W}\mathbf{G}\mathbf{S})_{ik}}{(\mathbf{G}\Lambda)_{ik}}\Big)^{\frac{1}{4}},\ \Lambda = \frac{\mathbf{S}\mathbf{G}^T\mathbf{W}\mathbf{G} + \mathbf{G}^T\mathbf{W}\mathbf{G}\mathbf{S}}{2}$$

where Λ is the Lagrangian multiplier to enforce $\mathbf{G}^T\mathbf{G} = I$.

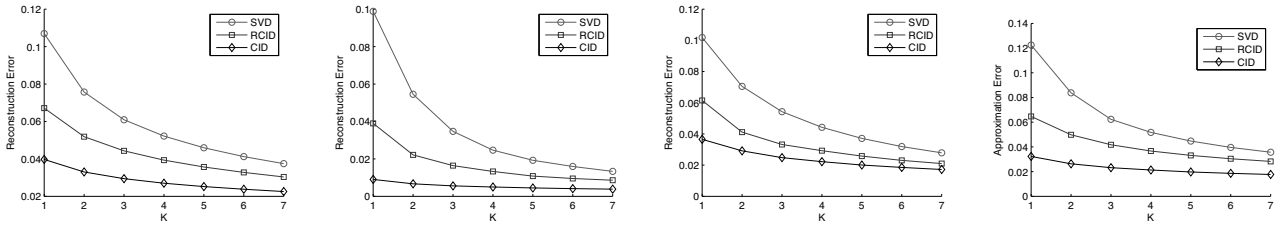
Repeat **(B1, B2)** until convergence. We obtain Relaxed CID result. After that, we fix the nonzero elements of $\mathbf{F}, \mathbf{G}$ to be 1, and update $\mathbf{S}$ to get CID result. Eq. (16) uses least square objective. A more sophisticated objective is using Laplacian formulism as presented in [Luo *et al.*, 2009].

The above algorithm assumes the input data $\mathbf{W}$ is nonnegative. When $\mathbf{W}$ has mixed signs, $\mathbf{S}$ has mixed signs. We have algorithm for updating $\mathbf{F}, \mathbf{G}, bS$ for this case. Details are skipped due to space limit.

The upper error bound of Theorem 1 can also be extended to this case. Because $\mathbf{W}$ is symmetric, only clustering of columns of $\mathbf{W}$ are done, which gives cluster indicator $\mathbf{G}_0$. Let $\mathbf{Y}_{\mathbf{W}}$ be constructed same as $\mathbf{Y}$ in Theorem 1, and evaluate $J^{\text{rows}}_{\text{Kmeans}}(\mathbf{Y}_{\mathbf{W}})$ using the cluster indicator $\mathbf{G}_0$. We have

Theorem 2.2 *In CID composition for input symmetric matrix* $\mathbf{W}$*, we have the upper bound for CID decomposition* $\mathbf{W} \cong (\mathbf{G}^*)\mathbf{S}^*(\mathbf{G}^*)^T$

$$\|\mathbf{W} - \mathbf{G}^*\mathbf{S}^*(\mathbf{G}^*)^T\|^2 \le J^{columns}_{Kmeans}(\mathbf{W}) + J^{rows}_{Kmeans}(\mathbf{Y}_{\mathbf{W}}). \tag{17}$$



(a) WANG's dataset (b) FERRET face database (c) Caltech 101 (d) VOC2009

Figure 2: Average reconstruction errors of CID, Relaxed CID (RCID), SVD on image compressions of four public computer vision datasets. When the rank of SVD increase from 1 to 7, we select the corresponding parameters for CID and Relaxed CID to make them use the same storage for compressed images.

3 Reconstruction Complexity of CID

One important application of matrix decomposition is that it helps to solve large scale problems due to storage reduction. Here, the speed of reconstruction of Kernel becomes a critical factor because we have to reconstruct the kernel very frequently. For example, in SVMs for classification.

We note that reconstruction in CID is extremely fast, as compared to other decomposition methods such as Nyström, SVD, CUR, etc. Specifically, we note that in CID, $\mathbf{F}$ is row cluster indicator and its role is to pick (index) the correct block means S_{kl}. There is **no computation** involved. The reconstruction of CID approximation is (supposing $\mathbf{X}$ is a kernel matrix $\mathbf{K}$),

$$\mathbf{K}_{ij} \approx (\mathbf{FSG}^T)_{ij} = \sum_{pq} \mathbf{S}_{pq}\mathbf{F}_{ip}\mathbf{G}_{jq} = \mathbf{S}_{R_i C_j}, \qquad (18)$$

where R_i is the row cluster that row index i belongs to, and C_j is the column cluster that column index j belongs to. We can obtain the kernel value by two indexing operations.

For other decomposition approaches, the reconstruction complexity is much higher. For example, in SVD approximation, $\mathbf{K}_{ij} \approx \sum_p \mathbf{U}_{ip}\mathbf{S}_{pp}\mathbf{U}_{jp}$ requires $3K_{\text{SVD}}$ floating point operations. For Nyström, it is even higher.

In an efficient implementation of Support Vector Machines, chunking techniques are widely employed. In chunking, we compute (reconstruct) kernel values for a chunk of the kernel matrix, use them for some computation, and throw them away. For the same chunk of the kernel matrix, if we later need, we recompute them anew. This is repeated many times. This is extremely expensive for the traditional approaches. But for CID, we access the kernel matrix values as if all the entries have been already computed and been stored.

4 Experimental Results

4.1 Reconstruction errors of CID

Image Compression

In order to examine CID low-rank approximation schemes on **rectangular** data matrices, we use image compression application and compare results to SVD that is considered as the best low-rank approximation method so far. First, we illustrate four image compression results using SVD and CID in Fig. 1. These four images are randomly selected from three public image databases: two from **WANG**'s dataset [Wang *et al.*, 2001], one from **Google** map, and one from **Ferret** face database [Philips *et al.*, 1998]. In order to compare them together, all images are resized to 384×256. The parameters are selected to make them use the same storage for compressed images. For color images (top row of Fig. 1), all methods are done independently on red, green, and blue channels. The reconstruction (low-rank approximation) errors are defined as $\|\mathbf{X}-\widetilde{\mathbf{X}}\|^2/\|\mathbf{X}\|^2$, where $\mathbf{X}$ is the original image and $\widetilde{\mathbf{X}}$ is the reconstruction image). Compared to SVD, in Fig. 1, CID and Relaxed CID provide not only more interpretable decomposition, but also much better low-rank approximation.

Besides the above illustrative demonstrations, we also systematically perform the image compression on four public computer vision datasets: **WANG**'s dataset [Wang *et al.*, 2001], **FERRET** face database [Philips *et al.*, 1998], **Caltech 101** [Perona *et al.*, 2004], and PASCAL Visual Object Classes 2009 Challenge (**VOC2009**)[1]. The experimental setup is the same as the above descriptions. When the rank of SVD increase from 1 to 7, we select the corresponding parameters for CID and Relaxed CID to make them use the same storage for compressed images. The average image reconstruction errors of SVD, CID, and Relaxed CID are plotted in Fig. 2. Using the same storage, CID always has the best low-rank approximation, and Relaxed CID is also consistently more accurate than SVD. We note that CID outperforms other methods the more structure the better and the at the smaller storage (small K_{SVD}) the better. On the other hand, for data without structure, CID does not outperform SVD.

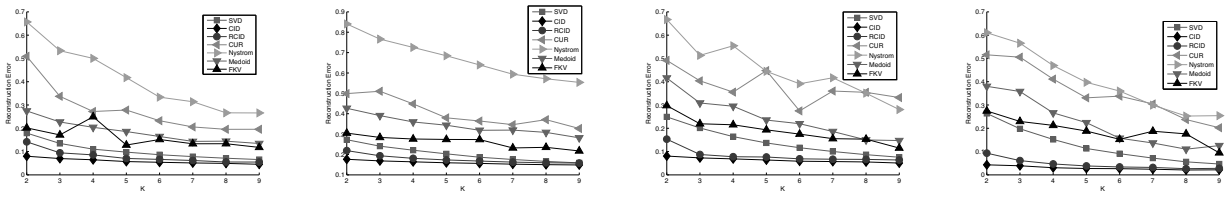
Kernel Matrix Approximation To measure CID low-rank approximation schemes on **symmetric/kernel** matrices, we perform CID and Relaxed CID on 3 commonly used human face datasets: AT&T face dataset[2] (Dim=644, N=400), MNIST digit images [Cun *et al.*, 1998] (Dim=784, N=150), and The Japanese Female Facial Expression (JAFFE) Database[3](Dim=4096, N=213), and one hand written digit-letter image database: BinAlpha[4] (Dim=320, N=1404). We perform methods on Gaussian kernel $\mathbf{K}(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/\gamma^2)$ (it is the standard one to evaluate the low-rank ap-

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/

[2] http://www.cl.cam .ac.uk/research/dtg/attarchive/facedatabase.html

[3] http://www.kasrl.org/jaffe.html

[4] http://www.cs.toronto.edu/ roweis/data.html



(a) AT&T (b) BinAlpha (c) MNIST (d) JAFFE

Figure 3: Approximation error comparisons of CID, Relaxed CID (RCID), Nyström decomposition, CUR, FKV, and SVD on kernel matrices computed from 4 public datasets.

proximation) where γ is the average of pairwise distances. We compare CID and Relaxed CID to SVD and major Nyström style decompositions on kernel matrices approximations. The results are shown in Fig. 3.

For the Nyström style decompositions, we run 20 random trials and obtain the average reconstruction errors. Four methods are compared in our experiments: (1) standard Nyström (Nystrom), (2) Medoid method (Medoid) [Zhang *et al.*, 2008], (3) Frieze *et al.*'s method [Frieze *et al.*, 2004] (FKV), (4) CUR. When the rank K of SVD changes from 2 to 9, parameters in all other methods are selected to guarantee the same storage is used. Obviously, our CID and Relaxed CID methods have much better low-rank approximation than all other related methods.

4.2 Kernel matrix approximation in classification

We embed the decomposition into kernel classification machines to investigate the efficiency. We compare CID to SVD and Nyström in terms of classification accuracy and kernel reconstruction time. We use RBF kernel Support Vector Machines (a modification of LIBSVM[5]) with 10-fold cross validation. We determine the parameters as following. By running 10-fold cross validation with gaussian parameter $\gamma = [2^{-10}\bar{r}, 2^{-9}\bar{r}, ..., 2^{10}\bar{r}]$, and C parameter $C = [2^{-10}, 2^{-9}, ..., 2^{10}]$ where $\bar{r}$ is the average Euclidean distance of all pairs of data points. Let (γ_{opt}, C_{opt}) denote the parameters which generate the best 10-fold cross validation results, then we perform decompositions on the kernel using gaussian parameter γ_{opt}: $\mathbf{K}^{opt}(i,j) = \exp(-\|x_i - x_j\|^2/\gamma_{opt}^2)$.

We chose $k = 1, 4, 8$ in our experiments. For Nyström and CID, we approximate the best kernel $\mathbf{K}^{opt}$ with the same size of storage as SVD. Notice that for CID, the kernel is reconstructed by using Eq. (18), which is extremely computationally simple.

We report results in Table 4.1. In LIBSVM, we use 1-vs-1 scheme, and reported results are derived by summing up all the base binary classifiers over all the cross validation folds (except the accuracy is the average accuracy). In Table 4.1, results show that CID outperforms SVD and Nyström in terms of both classification accuracy and kernel reconstruction CPU time. We also notice that in JAFFE dataset, even though we only use the same storage as $k = 1$ in SVD, the classification accuracy is still very high (96.64%), *i.e.* CID works well with very limited memory.

[5] www.csie.ntu.edu.tw/ cjlin/libsvm/

5 Conclusion

We proposed a family of algorithms to derive effective low-rank matrix decompositions which are both accurate and highly interpretable. In CID, column/row matrix factors are cluster indicators which can be compacted into very small space and the middle factor is the block cluster mean. Furthermore, the reconstruction is extremely fast. We derived several error bounds for the new matrices decomposition methods. Experiments on 4 computer vision data sets indicate CID provides better low-rank approximation than SVD at small subspace dimension. There could be many other ways to use clustering to generate low-rank matrix decompositions. Our approaches open a new application area for data clustering and efficiently solve the data low-rank approximation problem existing in many large-scale applications.

Acknowledgment This research is partially supported by NSF-CCF-0830780, NSF-DMS-0915228, NSF-CCF-0917274.

References

[Bach and Jordan, 2005] Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, volume 119, pages 33–40, 2005.

[Cun *et al.*, 1998] Y. L. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324, 1998.

[Ding *et al.*, 2006] Chris H. Q. Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, pages 126–135. ACM, 2006.

[Drineas and Kannan, 2003] P. Drineas and R. Kannan. Pass efficient algorithm for large matrices. In *SODA*, pages 223–232, 2003.

[Fowlkes *et al.*, 2004] Charless Fowlkes, Serge Belongie, Fan R. K. Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004.

[Frieze *et al.*, 2004] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

[Garg *et al.*, 2009] Rahul Garg, Hao Du, Steven M. Seitz, and Noah Snavely. The dimensionality of scene appearance. *ICCV*, 2009.

JAFFE		Accuracy	#SV	#Train	Train CPU	#Test	Test CPU
k=1	SVD	50.94	1804	241460	0.3204	76770	0.0446
	Nyström	69.44	1419	218641	0.4244	60396	0.1200
	CID	96.64	1114	233700	0.0458	47502	0.0122
k=4	SVD	88.44	814	132814	0.4764	34602	0.1345
	Nyström	82.42	1023	153943	1.7154	43496	0.0990
	CID	99.64	1125	226650	0.0553	47976	0.0225
k=8	SVD	98.67	662	118680	0.1734	28184	0.0354
	Nyström	86.26	877	142290	0.4749	37332	0.0916
	CID	99.64	1218	245435	0.0466	51810	0.0091
MNIST		Accuracy	#SV	#Train	Train CPU	#Test	Test CPU
k=1	SVD	47.00	1222	166168	0.5445	36160	0.0398
	Nyström	47.00	1198	199207	0.3957	35440	0.0719
	CID	76.50	839	155716	0.0505	24980	0.0169
k=4	SVD	77.00	861	116311	0.1504	25640	0.0309
	Nyström	61.00	1072	180348	0.3847	31800	0.0713
	CID	87.00	593	124770	0.0845	17700	0.0122
k=8	SVD	86.50	675	100396	0.1503	20220	0.0275
	Nyström	81.00	1162	186022	0.4742	34380	0.0849
	CID	93.00	644	124082	0.0841	19120	0.0149
BinAlpha		Accuracy	#SV	#Train	Train CPU	#Test	Test CPU
k=1	SVD	18.50	12406	16433484	13.589	3481488	2.1592
	Nyström	14.91	12607	18775496	45.030	3537792	6.8047
	CID	52.55	11574	17083114	1.150	3248496	0.220
k=4	SVD	37.18	11877	9665811	14.771	3333816	5.9979
	Nyström	24.28	12198	14881034	35.554	3423024	7.2677
	CID	62.11	8365	9601854	2.510	2347848	1.5901
k=8	SVD	59.84	11137	8467153	14.547	3126096	4.157
	Nyström	42.73	11556	10466382	28.136	3243528	7.668
	CID	64.38	9334	12096700	2.201	2619792	0.770
AT&T		Accuracy	#SV	#Train	Train CPU	#Test	Test CPU
k=1	SVD	39.00	3546	1120752	0.7608	283680	0.0666
	Nyström	32.75	3427	956909	1.9241	274160	0.5639
	CID	67.50	3481	1732176	0.1170	278480	0.0188
k=4	SVD	75.75	3021	783686	1.2742	241680	0.1931
	Nyström	58.50	3194	941741	2.3007	255520	0.7319
	CID	80.25	3600	2667600	0.680	288000	0.0195
k=8	SVD	95.50	3141	906251	1.2203	251280	0.2401
	Nyström	86.75	3116	900857	2.5544	249280	0.6228
	CID	97.25	1673	767862	0.5200	133840	0.0907

Table 1: Results of kernel reconstruction and classification using Support Vector Machines at different choices of k on **JAFFE, MNIST, BinAlpha** and **AT&T**. #SV: the total number of support vectors. #Train and #Test : total times of accessing the kernel in training or testing. TrainCPU and TestCPU: total CPU time (in seconds) of accessing the kernel in training or testing.

[Han and Kanade, 2001] Mei Han and Takeo Kanade. Multiple motion scene reconstruction from uncalibrated views. *ICCV*, 16:163–170, 2001.

[Lee and Seung, 1999] D.D. Lee and H.S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[Luo *et al.*, 2009] D. Luo, C. Ding, H. Huang, and T. Li. Non-negative Laplacian Embedding. In *2009 ICDM*, pages 337–346, 2009.

[Perona *et al.*, 2004] P. Perona, R. Fergus, and F. F. Li. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Workshop on Generative Model Based Vision*, page 178, 2004.

[Philips *et al.*, 1998] I. Philips, H. Wechsler, J. Huang, and P. Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing*, 16:295–306, 1998.

[Srebro and Jaakkola, 2003] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *ICML*, pages 720–727, 2003.

[Talwalkar *et al.*, 2008] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. *CVPR*, 2008.

[Turk and Pentland, 1991] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. *CVPR*, 1991.

[Wang *et al.*, 2001] James Ze Wang, Jia Li, and Gio Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture LIbraries. *IEEE Trans. Pattern Anal. Mach. Intell*, 23(9):947–963, 2001.

[Williams and Seeger, 2000] Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. pages 682–688. MIT Press, 2000.

[Zhang *et al.*, 2008] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved nyström low-rank approximation and error analysis. *ICML*, pages 1232–1239, 2008.

[Zhang *et al.*, 2009] Li Zhang, Sundeep Vaddadi, Hailin Jin, and Shree K. Nayar. Multiple view image denoising. *CVPR*, 2009.