A Divide-and-conquer Method for Scalable Low-rank Latent Matrix Pursuit

Yan Pan, Hanjiang Lai, Cong Liu
Sun Yat-sen University, Guangzhou, China
{panyan5,laihanj,liucong3}@mail.sysu.edu.cn

Shuicheng Yan
National University of Singapore, Singapore
eleyans@nus.edu.sg

Abstract

Data fusion, which effectively fuses multiple prediction lists from different kinds of features to obtain an accurate model, is a crucial component in various computer vision applications. Robust late fusion (RLF) is a recent proposed method that fuses multiple output score lists from different models via pursuing a shared low-rank latent matrix. Despite showing promising performance, the repeated full Singular Value Decomposition operations in RLF's optimization algorithm limits its scalability in real world vision datasets which usually have large number of test examples. To address this issue, we provide a scalable solution for large-scale low-rank latent matrix pursuit by a divide-and-conquer method. The proposed method divides the original low-rank latent matrix learning problem into two size-reduced subproblems, which may be solved via any base algorithm, and combines the results from the subproblems to obtain the final solution. Our theoretical analysis shows that with fixed probability, the proposed divide-and-conquer method has recovery guarantees comparable to those of its base algorithm. Moreover, we develop an efficient base algorithm for the corresponding subproblems by factorizing a large matrix into the product of two size-reduced matrices. We also provide high probability recovery guarantees of the base algorithm. The proposed method is evaluated on various fusion problems in object categorization and video event detection. Under comparable accuracy, the proposed method performs more than 180 *times faster than the state-of-the-art baselines on the CCV dataset with about 4,500 test examples for video event detection.*

1. Introduction

Effectively integrating multiple cues/features to obtain an accurate model, which is known as data fusion, is a popular approach to solve various tasks in computer vision and multimedia analysis, such as object classification and detection [4, 17, 19], video event detection [7, 8]. Various methods that fuse multiple kinds of features have been proposed in the literature (see, e.g. [18, 14, 4, 21], to just name a few).

These methods can be categorized into two main streams. The first stream is *Early Fusion*, which combines multiple features into a common representation before (or in) the learning process. Multiple Kernel Learning (MKL) [1] is one of the representative method in this stream. The second stream is *Late Fusion*, which firstly learns models from different kinds of features, and then combines the intermediate output scores of the learned models to yield a final model.

In this paper, we focus on the problem of Robust Late Fusion (**RLF**) proposed in [21], in which the authors formulated late fusion as an optimization problem of **L**ow-rank **L**atent **M**atrix **P**ursuit (**LLMP**). As shown in Figure 1(a), RLF fuses multiple models via seeking a shared low-rank comparison matrix. The formulation of LLMP is general and can also be applied to (with simple extensions) other important problems, such as rank aggregation [5] for information retrieval and collaborative filtering.

The authors in [21] solved the LLMP problem via the popular Augmented Lagrange Multiplier method [9] (Hereafter we call it RLF-ALM for the algorithm in [21]). The main drawback of RLF-ALM is that it needs repeated operations of full Singular Value Decomposition (SVD), whose time complexity is cubic w.r.t. the number of test examples. This limits RLF-ALM's scalability on large-scale datasets with thousands of test examples.

To address this issue, we develop a scalable method for large-scale LLMP. Our method has two building blocks: (1) Inspired by the recent advance in divide-and-conquer matrix factorization, we develop a divide-and-conquer approach for LLMP. As illustrated in Figure 1(b), our method divides the original LLMP problem into reduced-size subproblems, solves the subproblems via any base algorithm for LLMP, and then combines the results from the subproblems to obtain the final solution. Our theoretical analysis shows that with fixed probability, the proposed divide-and-conquer algorithm has recovery guarantees comparable to those of the base algorithm. (2) To further improve the scalability, we propose an efficient base algorithm for the

[0]Corresponding author: Cong Liu

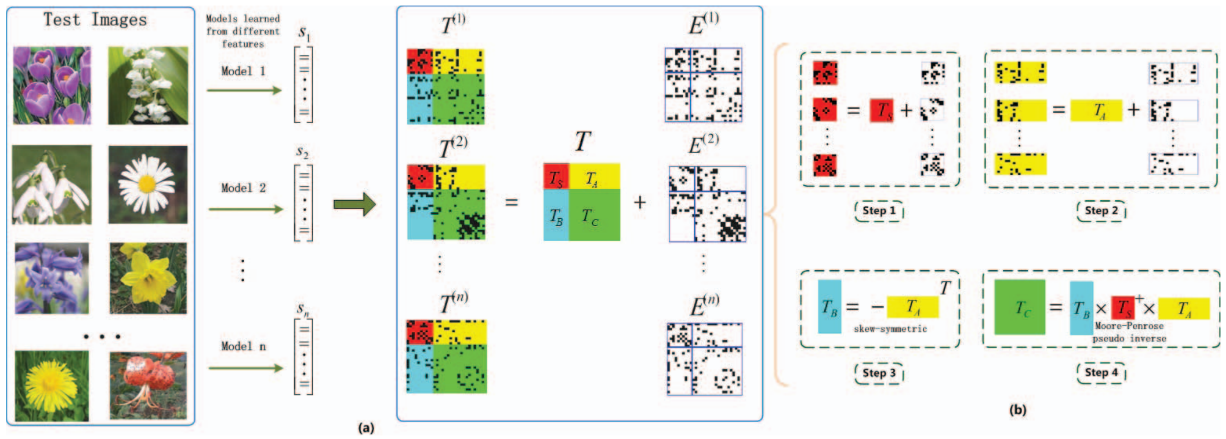

Figure 1. A divide-and-conquer method for Low-rank Latent Matrix Pursuit in Robust Late Fusion. **(a)** Overview of Robust Late Fusion. Given n score lists $s_i (i = 1 \ to \ n)$ from different models, RLF constructs n matrices $T^{(i)} (i = 1, 2, ..., n)$, where each $T^{(i)}$ encodes the pairwise relations of scores of every two test images in score list s_i. Then $T^{(i)} (i = 1, 2, ..., n)$ are used as input of LLMP to learn a shared, low-rank and skew-symmetric matrix T. Each $T^{(i)}$ can be reconstructed by T combining with an additive sparse matrix $E^{(i)}$. At last, RLF can recover a final and more accurate score list from T. **(b)** An illustration of the proposed divide-and-conquer method to solve the LLMP problem. We randomly divide T (and correspondingly, each $T^{(i)}$ and each $E^{(i)}$) into four parts T_S, T_A, T_B, T_C by sub-sampling. Then in *step 1-2*, we recover T_S and T_A by solving two size-reduced subproblems like the original LLMP problem, respectively. In *step 3*, we obtain T_B by skew-symmetry of T. In *step 4*, we calculate T_C by simple matrix algebra. Finally, we simply combine the four parts to obtain the final matrix T.

corresponding subproblems. In each iteration, the base algorithm factorizes a large comparison matrix $T \in \mathbb{R}^{m_1 \times m_2}$ into the product of two size-reduced matrices (i.e.,$m_1 \times r$ and $r \times m_2$, $r \ll min(m_1, m_2)$), so that on the one hand, the corresponding optimization problem only has $O((m_1 + m_2)r)$ variables; on the other hand, we only need to conduct a size-reduced SVD operation in each iteration, which is much cheaper than the SVD operation on the original full matrix. We further provide recovery guarantees of the proposed base algorithm. We conduct extensive experiments to evaluate the proposed method. The results on the Columbia Consumer Video (CCV) dataset with 4,658 test examples show that, under comparable accuracy, the propose method performs more than 180 times faster than the state-of-the-art RLF-ALM algorithm.

2. Related Work

Integrating multiple kinds of complementary features is a common approach in various vision tasks. The strategies of feature combination can be divided into early fusion and late fusion.

A representative method of early fusion is MKL [1], which simultaneously learns kernel matrices and their associated combination weights. Combining multiple features via MKL showed good performance in object classification [4] and object detection [19]. However, some empirical studies [4] also showed that MKL may not perform better than the simple average kernel combination.

Many methods of late fusion have been proposed. Nandakumar *et al.* [14] used finite Gaussian mixture to model the distribution of the intermediate output scores, and then combined the scores via likelihood ratio test. Terrades *et al.* [18] used a non-Bayesian probabilistic framework to fuse multiple output scores by minimizing the misclassification rates under the ℓ_1 constraint. The most related work to ours is the recently proposed Robust Late Fusion in [21], where the authors formulated late fusion as a problem of pursuing a shared low-rank latent matrix, and then proposed to the corresponding optimization problem via an algorithm based on Augmented Lagrange Multiplier method [9]. However, the optimization algorithm in [21] needs repeated full SVD operations whose time complexity is cubic w.r.t. the number of text examples, which limits its scalability on large-scale real world applications.

Our work is motivated by the recent progress in efficient algorithms for large-scale low-rank matrix learning. Mackey *et al.* [13] proposed a divide-and-conquer algorithm for robust principal component analysis (RPCA) [3] problems. The main idea is to divide a large-scale matrix learning problem into several size-reduced subproblems, solve the subproblems in parallel, and then combine the results from the subproblems. With fixed probability, their divide-and-conquer algorithm has estimation error bound guarantees compared to its base algorithm that directly solves the original large-scale problem. In contrast to RPCA with a single low-rank and sparse decomposition constraint, our work focus on a divide-and-conquer approach for the LLMP problem that seeks to learn a shared skew-symmetric comparison matrix with multiple low-rank and sparse decomposition constraints.

3. Problem Formulation

The robust late fusion problem [21] is to fuse n confidence score lists $\{s^{(i)}\}_{i=1}^n$. Each list $s^{(i)} = (s_1^{(i)}, s_2^{(i)}, ..., s_m^{(i)})^T$ is an intermediate output of a learned model (e.g., a classifier) from a specific kind of features/cues.

Next, we will first introduce how to construct the comparison matrices (as inputs to the LLMP problem), and then describe the formulation of LLMP.

For each $s^{(i)}$, we convert it into a $m \times m$ comparison matrix $T^{(i)}$, in which each $T_{j,k}^{(i)}$ is defined as:

$$T_{j,k}^{(i)} = sign(s_j^{(i)} - s_k^{(i)}). \tag{1}$$

Each comparison matrix $T^{(i)}$ is an isotonic representation that encodes the relative relationship among the test examples. Some entries in $T^{(i)}$ correctly reflect the relative relationship among the test examples, while other entries may be incorrect due to the wrong predictions in the score list $s^{(i)}$. $T^{(i)}$ can be viewed as a combination of two parts: a shared low-rank part T that reflects the correct relationship among the test examples, and a sparse part $E^{(i)}$ that encodes the irregular corruptions made by the incorrect entries. $T^{(i)}$ will be used as inputs to the LLMP optimization problem.

Ideally, assume there exists a score list s that correctly explains the relations of the test examples, then T can be viewed as a matrix which encodes the relations in s, i.e., $T = se^T - es^T$ where e denotes the vector with all ones. Obviously, T in this form is a rank-2 matrix. However, if there is large variation in the scores, the matrix T might have an unknown rank higher than 2. This motivates us to formulate RLF as an optimization problem of pursuing a shared, low-rank and skew-symmetric latent matrix:

$$\begin{aligned} &\min_{T,\{E^{(i)}\}_{i=1}^n} rank(T) + \lambda \sum_{i=1}^n ||E^{(i)}||_0 \\ &s.t.\ T^{(i)} = T + E^{(i)},\ i = 1, 2, ..., n, T = -T^T. \end{aligned} \tag{2}$$

It is known that (2) is NP-hard in general due to the non-convex $rank(T)$ and $||E^{(i)}||_0$ norm. One popular way is to replace $rank(T)$ with $||T||_*$, $||E^{(i)}||_0$ with $||E^{(i)}||_1$, which leads to the following LLMP optimization problem:

$$\begin{aligned} &\min_{T,\{E^{(i)}\}_{i=1}^n} ||T||_* + \lambda \sum_{i=1}^n ||E^{(i)}||_1 \\ &s.t.\ T^{(i)} = T + E^{(i)},\ i = 1, 2, ..., n, T = -T^T, \end{aligned} \tag{3}$$

where $||T||_*$ denotes the trace norm of T.

Given the learned matrix T, we can easily obtain the score list s by $s = \frac{1}{m} Te^T$.

The LLMP optimization problem is the core in RLF. More importantly, LLMP is general and can also be applied to other family of problems, such as rank aggregation [5] for information retrieval and collaborative filtering. In this paper, we only focus on its applications in computer vision tasks.

4. A Divide-and-Conquer Solution

The existing algorithms for LLMP, such as RLF-ALM [21], need repeated SVD operations on the whole comparison matrix, which have cubic time complexity w.r.t. the number of test examples. This makes them scale poorly on large-scale real world datasets. Inspired by the recent advance in divide-and-conquer algorithms [13] for RPCA problems [3], we propose a scalable divide-and-conquer method for the LLMP optimization problem. The main idea is to divide the optimization problem (3) into small-size subproblems, each of which can be easily solved by a base algorithm or by simple algebra, and then combine the results to get a low-rank and skew-symmetric comparison matrix.

The proposed divide-and-conquer method for LLMP is summarized in Algorithm 1. It has the following three main steps:

(1) Partitioning the matrices Let $T \in \mathbb{R}^{m \times m}$, and S be a set of k ($k \ll m$) indices randomly sampled from $\{1, 2, ..., m\}$ without replacement. For a matrix M, we denote M_S as the $k \times k$ submatrix in M where both the row set and column set are S. For ease of presentation, without loss of generality, we assume $S = \{1, 2, ..., k\}$, and then M_S is the top and left $k \times k$ submatrix of M. We sample the submatrices T_S, $T_S^{(i)}$ and $E_S^{(i)}$, and then each matrix is partitioned into four parts:

$$\mathbf{T} = \begin{bmatrix} T_S & T_A \\ T_B & T_C \end{bmatrix}, \tag{4}$$

$$\mathbf{T^{(i)}} = \begin{bmatrix} T_S^{(i)} & T_A^{(i)} \\ T_B^{(i)} & T_C^{(i)} \end{bmatrix}, \mathbf{E^{(i)}} = \begin{bmatrix} E_S^{(i)} & E_A^{(i)} \\ E_B^{(i)} & E_C^{(i)} \end{bmatrix}.$$

Since T is skew-symmetric, we have $T_A = -T_B^T$.

(2) Recovering the seed matrix We recover the seed matrix T_S by solving the following subproblem:

$$\begin{aligned} &\min_{T_S,\{E_S^{(i)}\}_{i=1}^n} ||T_S||_* + \lambda \sum_{i=1}^n ||E_S^{(i)}||_1 \\ &s.t.\ T_S^{(i)} = T_S + E_S^{(i)},\ i = 1, 2, ..., n, T_S = -T_S^T. \end{aligned} \tag{5}$$

Obviously this is a size-reduced problem of (3) which can be solved by any base algorithm for LLMP (i.e., RLF-ALM).

Algorithm 1: Divide-and-Conquer LLMP (DC-LLMP)

```
Input: $T^{(i)}$ (i=1,2,...,n), $\lambda$, $k$
Output: $T$, $T^{(i)}$ (i=1,2,...,n)
1. Randomly sample a set of $k$ indices to partition each
   matrix of $T^{(i)}$ into 4 parts $(T_S^{(i)}, T_A^{(i)}, T_B^{(i)}, T_C^{(i)})$;
2. Recover the seed matrix $T_S$ and $E_S^{(i)}$ by (5);
3. Recover $T_A$ and $E_A^{(i)}$ by (6);
4. Calculate $T_B$ and $E_B^{(i)}$ by (7);
4. Calculate $T_C$ and $E_C^{(i)}$ by (8);
5. Output: $T$, $E^{(i)} (i = 1, 2, ..., n)$.
```

(3) Recovering the rest parts We obtain $(T_A, E_A^{(i)})$ by solving the following subproblem:

$$\min_{T_A, E_A^{(i)}} ||T_A||_* + \lambda \sum_{i=1}^{n} ||E_A^{(i)}||_1 \quad s.t.\ T_A^{(i)} = T_A + E_A^{(i)}, i = 1, 2, ..., n, \tag{6}$$

Since T is skew-symmetric, we have $T_A = -T_B^T$. Hence, we can get $(T_B, E_B^{(i)})$ by

$$T_B = -T_A^T,\ E_B^{(i)} = -{E_A^{(i)}}^T \ (i = 1, 2, ...n). \tag{7}$$

Now we are ready to calculate the last part T_C by

$$T_C = T_B T_S^{+} T_A,\ E_C^{(i)} = T_C^{(i)} - T_C \ (i = 1, 2, ...n), \tag{8}$$

where $T_S^{+} = V\Sigma^{-1}U^T$ denote the pseudo inverse of T_S, and $U\Sigma V^T = T_S$ is the SVD form of T_S.

Remark The proposed method method can be viewed as a variant of Generalized Nystrom Method. The standard Nystrom method is used to accelerate large-scale learning problems with symmetric positive semi-definite matrices (i.e., kernel matrices learning) [20], and it has been generalized for arbitrary matrices [6]. It is worth noting that, although our framework is based on Generalized Nystrom Method, the involved subproblems (i.e., (5) and (6) with joint low-rank and sparse constraints) and their corresponding theoretical analysis are specific to LLMP, which are different from other work [13] using Generalized Nystrom Method.

5. Efficient Base Algorithm for Subproblems

The optimization problems (6) and (5) are essentially in the same form:

$$\min_{T, E^{(i)}} ||T||_* + \lambda \sum_{i=1}^{n} ||E^{(i)}||_1 \quad s.t.\ T^{(i)} = T + E^{(i)}, i = 1, 2, ..., n, \tag{9}$$

where $T,\ T^{(i)},\ E^{(i)} \in \mathbb{R}^{m_1 \times m_2}$.

Algorithm 2: Base Algorithm for LLMP (LLMP-Base)

```
Input: $\{T^{(i)}\}_{i=1}^n$, $\lambda$, $r$
Output: $T$, $\{E^{(i)}\}_{i=1}^n$
1. Initialize: $Q_1 = 0$, $J_1 = 0$, $E_1^{(i)} = 0$, $Y_1^{(i)} = 0$,
   $\mu_1 = 10^{-3}$, $\max_\mu = 10^{10}$, $\epsilon = 10^{-8}$, $\rho = 1.9$, $t = 0$.
2. Repeat
3.   $t \leftarrow t + 1$
4.   Fix the other terms and update $Q$ by
     $Q_{t+1} \leftarrow arg\min_{Q^TQ=I} \sum_{i=1}^n \frac{\mu_t}{2}||T^{(i)} - E_t^{(i)} - QJ_t||_F^2 + \sum_{i=1}^n \langle Y_t^{(i)}, T^{(i)} - E_t^{(i)} - QJ_t\rangle$;
5.   Fix the other terms and update $J$ by
     $J_{t+1} \leftarrow arg\min_J \sum_{i=1}^n \frac{\mu_t}{2}||T^{(i)} - E_t^{(i)} - Q_{t+1}J||_F^2 + ||J||_* + \sum_{i=1}^n \langle Y_t^{(i)}, T^{(i)} - E_t^{(i)} - Q_{t+1}J\rangle$;
6.   Fix the other terms and update each $E^{(i)}$ by
     $E^{(i)} \leftarrow arg\min_{E^{(i)}} \frac{\mu_t}{2}||T^{(i)} - E^{(i)} - Q_{t+1}J_{t+1}||_F^2 + \lambda||E^{(i)}||_1 + \langle Y_t^{(i)}, T^{(i)} - E^{(i)} - Q_{t+1}J_{t+1}\rangle$;
7.   update the multiplier
     $Y_{t+1}^{(i)} \leftarrow Y_t^{(i)} + \mu_t(T^{(i)} - E_{t+1}^{(i)} - Q_{t+1}J_{t+1})$;
8.   update the parameter $\mu_{t+1} \leftarrow \max(\max_\mu, \rho\mu_t)$;
9. until $||T^{(i)} - E_{t+1}^{(i)} - Q_{t+1}J_{t+1}||_\infty \leq \epsilon$
10. Output: $T_{t+1} = Q_{t+1}J_{t+1}$, $\{E_{t+1}^{(i)}\}_{i=1}^n$.
```

Although there exist algorithms to solve (9) (i.e., the algorithm in [21]), a challenging issue arising in solving (9) is that, when applied to subproblems with large m_1 and m_2, the repeated full SVD operations are still computationally expensive. To address this issue, we propose to factorize the large comparison matrix $T \in \mathbb{R}^{m_1 \times m_2}$ into the product of two size-reduced matrices (i.e.,$m_1 \times r$ and $r \times m_2$, $r \ll min(m_1, m_2)$). This kind of fixed rank prior is used in recent works [12, 11] to speed up low-rank matrix learning problems such as RPCA [3] and LRR [10]. There are two advantages from the factorization: (1) the resulting optimization problem only has $O((m_1+m_2)r)$ variables, and (2) only size-reduced SVD operations are needed in each iteration, which is much cheaper than the full SVD operations.

Our algorithm is based on the ALM framework [9]. We assume the factorization be $T = QJ$ where $Q \in m_1 \times r$ is column-orthogonal and $J \in r \times m_2$. The corresponding Lagarange function of (9) is:

$$\begin{aligned} L(Q, J, E^{(i)}, Y^{(i)}) = ||QJ||_* + \lambda \sum_{i=1}^{n} ||E^{(i)}||_1 \\ + \sum_{i=1}^{n} \frac{\mu}{2}||T^{(i)} - E^{(i)} - QJ||_F^2 \\ + \sum_{i=1}^{n} \langle Y^{(i)}, T^{(i)} - E^{(i)} - QJ\rangle. \end{aligned} \tag{10}$$

Algorithm 2 shows the sketch of the proposed algorithm.

Next we investigate the update rules for each variable (Q, J, $E^{(i)}$) when other variables being fixed.

Since Q is column-orthogonal, we have that $||QJ||_* = ||J||_*$. The optimization problem w.r.t. Q is:

$$\min_{Q^TQ=I} \frac{\mu}{2}\sum_{i=1}^{n}||T^{(i)} - E^{(i)} - QJ + \frac{Y^{(i)}}{\mu}||_F^2,$$

whose solution is $Q = U_Q V_Q^T$ where $U_Q\Sigma_Q V_Q^T$ is the SVD form of $\frac{1}{n}\sum_{i=1}^{n}(T^{(i)} - E^{(i)} + \frac{Y^{(i)}}{\mu})J^T$.

The optimization problem w.r.t. J is:

$$\min_J ||J||_* + \frac{\mu}{2}\sum_{i=1}^{n}||T^{(i)} - E^{(i)} - QJ + \frac{Y^{(i)}}{\mu}||_F^2.$$

This can be solved by singular value threshold method [2]. Its solution is $J = U_J\mathbb{S}_{\frac{1}{\mu}}(\Sigma_J)V_J^T$ where $U_J\Sigma_J V_J^T$ is the SVD form of $\frac{1}{n}\sum_{i=1}^{n}Q^T(T^{(i)} - E^{(i)} + \frac{Y^{(i)}}{\mu})$ and $\mathbb{S}_\gamma(X) = \max(0, X-\gamma) + \min(0, X+\gamma)$ denote the shrinkage operator [9].

The optimization problem w.r.t. $E^{(i)}$ is

$$\min_{E^{(i)}} \frac{\mu}{2}||T^{(i)} - E^{(i)} - QJ + \frac{Y^{(i)}}{\mu}||_F^2 + \lambda||E^{(i)}||_1,$$

whose solution is $E^{(i)} = \mathbb{S}_{\lambda/\mu}(T^{(i)} - QJ + \frac{Y^{(i)}}{\mu})$.

6. Theoretical Analysis

A natural question arising here is whether there is any theoretical guarantee on the recovery errors of the proposed divide-and-conquer method compared to the errors of the base algorithm. The answer is yes as we will show in the following theorems. Our analysis is based on the standard matrix coherence assumption. We refer the reader to Section 3.1.1 in [13] for detailed definitions of (μ, r)-coherence.

We first rewrite (9) to a more general form:

$$\min_{T,E^{(i)}} ||T||_* + \lambda\sum_{i=1}^{n}||E^{(i)}||_1 \quad (11)$$
$$s.t.\ ||T^{(i)} - T - E^{(i)}||_F \leq \Delta, i = 1, 2, ..., n.$$

Obviously, (11) reduces to (9) when $\Delta \to 0$.

The first theorem establishes that, under certain conditions, the solution to (11) guarantees low recovery errors with high probability.

Theorem 1. *Let $(T_0, E_0^{(i)})$ be the ground truth of the joint low-rank and sparse decomposition of $T^{(i)}$. Let $(T, E^{(i)})$ be the minimizer of (11). Suppose that T_0 is (μ_0, r_0)-coherent and T is (μ_T, r_T)-coherent. Suppose that $\forall i$, the support set of $E_0^{(i)}$ is uniformly distributed among all sets with cardinality $s_{0,i}$, and the support set of $E^{(i)}$ is uniformly distributed among all sets with cardinality $s_{T,i}$. Then if $m_1 \leq m_2$ and $\forall i$, $||T^{(i)} - T_0 - E_0^{(i)}||_F \leq \Delta$, there is a constant c_p such that with probability at least $1-2nc_p m_2^{-\beta}$, the minimizer $(T, E^{(i)})$ of (11) with $\lambda = 1/\sqrt{m_2}$ satisfies*

$$||T - T_0||_F^2 + \frac{1}{n}\sum_{i=1}^{n}||E^{(i)} - E_0^{(i)}||_F^2 \leq 2c_\epsilon^2 m_1 m_2\Delta^2$$

provided that $r_0 \leq p_r m_1\mu^{-1}log^{-2}(m_2)$, $r_T \leq p_r m_1\mu^{-1}log^{-2}(m_2)$, $\forall i$, $s_{0,i} \leq (1-p_s\beta)m_1m_2$ and $s_{T,i} \leq (1-p_s\beta)m_1m_2$ for $\beta \geq 2$ and some positive constants p_r, p_s, c_ϵ.

The proof is provided in the supplementary material.

The second theorem shows that, with fixed probability, the recovery errors of the proposed divide-and-conquer method is bounded by the errors of the solution to (11).

Theorem 2. *Let T_0 be the ground truth of the rank-r_0 latent matrix, and T be the output of Algorithm 1. T is partitioned as in (4), and correspondingly T_0 is partitioned into $(T_{0,S}, T_{0,A}, T_{0,B}, T_{0,C})$. Choose $k \geq c\mu r log(m_2)log(1/\delta)/\epsilon^2$, with c being a fixed positive constant, under the notations of Algorithm 1, with probability at least $(1-\delta)(1-\delta-0.2)$, both $C_0 = [T_{0,S}\ T_{0,A}]$ and $R_0 = [T_{0,S}^T\ T_{0,B}^T]^T$ are $(\frac{r_0\mu_0^2}{1-\epsilon/2}, r_0)$-coherent and T satisfies*

$$||T - T_0||_F \leq (2+3\epsilon)\sqrt{||T_S - T_{0,S}||_F^2 + 2||T_A - T_{0,A}||_F^2}.$$

The proof is provided in the supplementary material.

7. Experimental Evaluation

In this section, we evaluate the accuracy and running time of the proposed divide-and-conquer method on various real world and simulated datasets. We compare the performance of three algorithms for the LLMP problem, including the proposed base algorithm LLMP-Base, the proposed divide-and-conquer algorithm DC-LLMP, and the state-of-the-art RLF-ALM[1] algorithm [21]. All the experiments were conducted with Matlab R2010b on an Apple PC with 8G memory and 2.5Ghz i5-2400S CPU.

[1]Since the code of the algorithm in [21] is not publicly available, we carefully implemented RLF-ALM. Note that the original RLF-ALM algorithm in [21] enforces the learned low-rank matrix to be rank-2 as a convergence condition. As explained in Section 3, in practice, the matrix T might have an unknown rank higher than 2 due to large variation in scores. Hence, in our implementation of RLF-ALM, we relax the algorithm by removing the rank-2 constraint from the convergence conditions. We found in experiments that with or without the rank-2 constraint, RLF-ALM has nearly the same accuracy. In addition, removing such a convergence condition would not slow down the algorithm because it would only be more easy to reach the convergence conditions.

7.1. Experiments on real world datasets

We evaluate the proposed method on object categorization and video event detection tasks. For a fair comparison, we follow the settings in [21]. We use one-vs-all SVM to generate the output score lists from different kinds of features. We use Mean Average Precision (MAP) over all categories in the dataset as the evaluation metric. For SVM, we use χ^2 kernel which is calculated by $exp(-\frac{1}{\sigma}d_{\chi^2}(x,y))$ where σ is set as the average value of all pairwise distances on the training set. The tradeoff parameter C in SVM and the regularization parameter λ in LLMP are chosen from $\{10^{-3}, 10^{-2}, ..., 10^{3}\}$ by cross validation[2]. For the algorithms (RLF-ALM, LLMP-Base and DC-LLMP) based on the ALM framework, we initialize the parameters $\mu_1 = 10^{-3}$ and set $\rho = 1.9$. In all the experiments, the running time results of DC-LLMP is the average of 5 trials.

To compare MAP, we include the results of three additional early and late fusion baselines: (1) kernel average, an early method which averages multiple kernel matrices to a single kernel matrix for learning; (2) SimpleMKL [16], a representative of Multiple Kernel Learning; (3) average late fusion, which obtains the final model by directly averaging the intermediate output scores from different models. For convenience, some results of these baselines are directly cited from [21].

Table 1. MAP and running time comparison results on Oxford Flower 17 dataset.

Method	MAP	Time (seconds)
Kernel Average	0.860± 0.0017	-
SimpleMKL	0.863± 0.0021	-
Average Late Fusion	0.869± 0.0021	-
RLF-ALM	0.903± 0.0019	8.75
Our LLMP-Base	0.903± 0.0017	4.02
Our DC-LLMP	0.902± 0.0024	0.92

Results on object categorization In this section, we present the results on object categorization. We use the Oxford Flower17 dataset [15] which has various flower images from 17 categories. Each category has 80 samples. In our experiments, we directly use the pre-defined three splits in the dataset (with 680/340/340 training/validation/test examples, respectively) and the corresponding pre-computed distance matrices. These distance matrices are computed from seven kinds of features: HOG, color, shape, texture, clustered HSV values, SIFT on the foreground internal region (SIFTint) and on the foreground boundary (SIFTbdy) (see [15] for details).

We simply set $r = 20$ in LLMP-Base and DC-LLMP, $k = 50$ in DC-LLMP. The running time and MAP are first averaged across all 17 categories and then averaged over the three splits.

As shown in Table 1, three observations can be made from the comparison results: (1) The proposed DC-LLMP and LLMP-Base have comparable MAP to RLF-ALM. And all the three robust late fusion methods have superior performance on MAP over the other three early and late fusion baselines. In particular, the proposed DC-LLMP outperforms Kernel Average, SimpleMKL, Average Late Fusion by 4.5%, 4.9% and 3.8% relatively. (2) The proposed LLMP-Base shows near-linear speed-up (i.e. more than 2 times) over RLF-ALM. (3) The proposed DC-LLMP performs almost an order of magnitude (i.e., 9.51 times) faster than RLF-ALM.

Results on video event detection We evaluate the proposed method on video event detection using the Columbia Consumer Video dataset [7] which contains 9,317 video over 20 classes of semantic events. The dataset is divided into a training set with 4,659 videos and a test set with 4,658 videos. The dataset also provides three kinds of Bag-of-Words (BOW) features, including 5,000 dimensional SIFT BOW features, 4,000 dimensional Mel-frequency cepstral coefficients BOW features and 5,000 spatial-temporal interest points BOW features. We directly use the provided features to construct the χ^2 kernels.

Table 2 shows the comparison results on the CCV dataset. The results indicate that: (1) The proposed LLMP-Base and DC-LLMP have comparable performance to RLF-ALM in terms of MAP. And the three methods for RLF outperforms the other three baselines. For instance, the proposed DC-LLMP has relatively 4.4%, 3.9% and 4.8% performance gain over Kernel Average, SimpleMKL and Average Late Fusion. These results are consistent with those on object categorization. (2) Under comparable MAP, DC-LLMP performs more than 30 times faster than its base algorithm LLMP-Base, and it performs more than 180 times faster than the state-of-the-art RLF-ALM. We can conclude that DC-LLMP has superior running time performance in real world vision tasks which usually have thousands of test examples, and it is a practical and scalable method for large-scale low-rank latent matrix pursuit.

Table 2. MAP and running time comparison results on Columbia Consumer Video dataset.

Method	MAP	Time (seconds)
Kernel average	0.597	-
SimpleMKL	0.603	-
Average late fusion	0.595	-
RLF-ALM	0.6249	8447
Our LLMP-Base	0.6241	1386
Our DC-LLMP	0.6235	46

Effects of the parameters r and k are two important parameters in LLMP-Base and DC-LLMP. An implicit as-

[2]Although we can simply set $\lambda = 1/\sqrt{m_2}$ by Theorem 1, we found that choosing λ by cross validation leads to better performance.

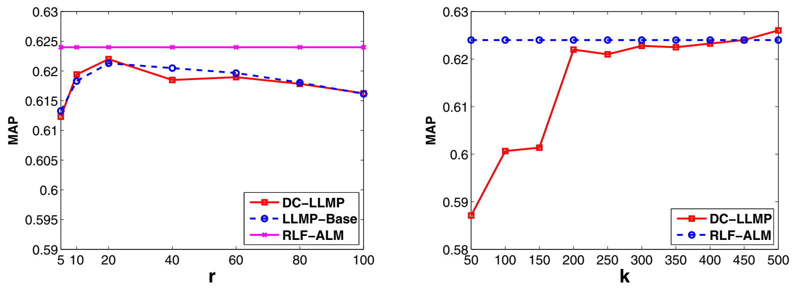

Figure 2. Effects of parameters on CCV dataset. (a) effects on MAP w.r.t. r. (b) effects on MAP w.r.t. k.

sumption in the design of LLMP-Base and DC-LLMP is that the parameters r and k should be much smaller than the corresponding input size (i.e., $r \ll \min(m_1, m_2)$ and $k \ll m$), which is a key factor to speed up the algorithms. Hence, in practice, we need to set r and k to be relatively small values. A natural question arising here is whether these relatively small r and k would degrade the performance of LLMP-Base and DC-LLMP. To answer this question, We conducted experiments on the CCV dataset to observe the effects of different r and k.

Figure 2(a) shows the MAP results of DC-LLMP and LLMP-Base with different values of r when fixing $k = 200$ and λ being the corresponding best value of each category (chosen from cross validation). The results in Figure 2(b) indicate that when $r \geq 10$, DC-LLMP and its base algorithm LLMP-Base have comparable MAP values to those of RLF-ALM. Moreover, DC-LLMP and LLMP-Base seem to be insensitive to r as long as r is in a suitable range (i.e. $10 \leq r \leq 50$).

Figure 2(b) shows the MAP results of DC-LLMP with different values of k when fixing $r = 20$ and λ being the corresponding best value of each category (chosen from cross validation). We can observe from Figure 2(a) that when k is not too small (i.e.,$k > 200 \approx m/20$), the MAP of DC-LLMP is nearly the same as its base algorithm LLMP-Base and the RLF-ALM baseline. This justifies that DC-LLMP has comparable performance with the algorithms directly working on full matrices under relative small values of k, which is a key factor that leads to the speed-ups of DC-LLMP.

7.2. Simulation results

To further investigate the characteristics of the proposed algorithms, we conduct simulated experiments to observe the effects on the running time and MAP with (1) different input data size (i.e., different values of m and n), and (2) different proportions of corrupted data entries.

In our simulation, we firstly generate n score lists $f_1, f_2, ..., f_n$, each of which contains m scores randomly sampled from $Normal(0, 1)$. Then we add random noise (sampled uniformly from $[-100, 100]$) to $s\%$ entries of each score list. We set the top 20% examples with largest average scores as positive examples and the rest as negative. After that, we use each score list to construct the corresponding input matrix $T^{(i)}$. In all simulations, we fix $\lambda = 100$.

Effects of different input data size The first simulation is to explore the effects on the running time w.r.t. different number of test examples (m) and different number of models (n). In practice, we usually assume matrix T have small rank r_0, and set k to be $r_0 + p$ with a small constant p. Hence, in this simulation, we fix $r_0 = 10$ and $k = 20$

Figure 4(a) shows the comparison results on running time with different values of m when other parameters being fixed. We set $n = 5$. Two observation can be made from the results: (1) LLMP-Base has nearly linear speed-up compared to RLF-ALM. As m increases, the speed-up also increases. (2) When m is relative small, DC-LLMP shows linear speed-up compared to LLMP-Base and RLF-ALM. When m is large, DC-LLMP shows super-linear speed-up compared to LLMP-Base and RLF-ALM.

Figure 4(b) shows the comparison results on running time with different values of n. We set $m = 1000$. The results indicate that: (1) The running time of all the three algorithms slowly increase with n increasing. (2) For the speed of time increasing, we have RLF-ADM $>$ LLMP-Base $>$ DC-LLMP.

Effects of different proportion of corruptions Figure 4(c) show the comparison results on MAP with different proportion of corrupted entries in each score list, assuming other parameters being fixed. We set $m = 1000$, $n = 5$. The results show that, under different percentage of noisy entries, DC-LLMP has similar performance compared to LLMP-Base and RLF-ALM.

8. Conclusions

In this paper, we developed DC-LLMP, a scalable divide-and-conquer method for large low-rank latent matrix pursuit. DC-LLMP divides the original low-rank latent matrix learning problem into size-reduced subproblems which can be solved cheaply by a base algorithm, and obtains the fi-

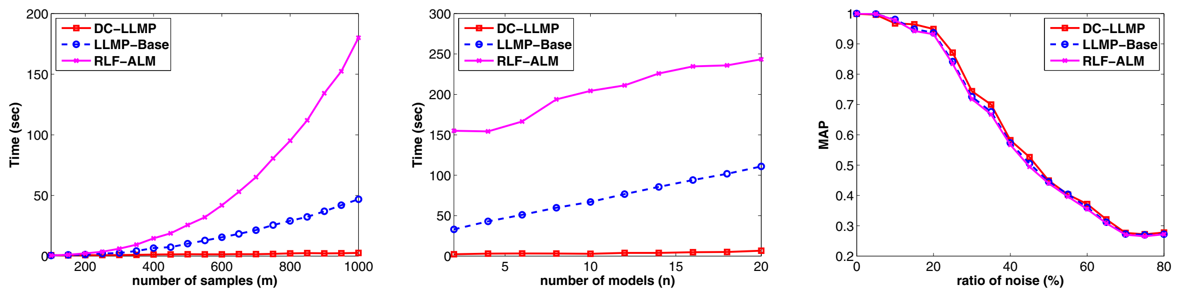

Figure 3. Comparison results on simulated data. (a) effects on running time w.r.t. m. (b) effects on running time w.r.t. n. (c) effects on MAP w.r.t. different percentage of noise.

nal results by combining the results from the subproblems. The theoretical analysis showed that the recovery errors of DC-LLMP are bounded by the recovery errors of the base algorithm with fixed probability. Furthermore, we developed an efficient base algorithm to solve the corresponding subproblems. Extensive empirical evaluations showed the superior efficiency of LLMP over the state-of-the-art baselines.

Acknowledgment

This work was funded in part by National Science Foundation of China (grant No. 61003045, 61003241, 61033010), Natural Science Foundation of Guangdong Province, China (grant No. 10451027501005667), Educational Commission of Guangdong Province, China, Shenzhen Special Development Fund for Emerging Strategic Industries (grant No. JCYJ20120615151826623), and the Fundamental Research Funds for the Central Universities.

References

[1] F. Bach, G. Lanckriet and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. *In ICML*, 2004. 1, 2

[2] J-F. Cai, E. J. Candes and Z. Shen. A singular value thresholding algorithm for matrix completion. *UCLA CAM Report*, 2008. 5

[3] E. Candes, X. Li, Y. Ma and J. Wright. Robust principal component analysis? *JACM*, 58(1):1-37, 2009. 2, 3, 4

[4] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. *In ICCV*, 2009. 1, 2

[5] D. F. Gleich and L. H. Lim. Rank aggregation via nuclear norm minimization. *In KDD*, 2011. 1, 3

[6] S. Goreinov, E. Tyrtyshnikov and N. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1-3):1-21, 1997. 4

[7] Y-G. Jiang, G. Ye, S-F. Chang, D. Ellis and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. *In ICMR*, 2011. 1, 6

[8] Z-Z. Lan, L. Bao, S-I. Yu, W. Liu and A. G. Hauptmann. Double Fusion for Multimedia Event Detection. *In MMM*, 2012. 1

[9] Z-C. Lin, M-M. Chen and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. arxiv:1009.5055. 1, 2, 4, 5

[10] G-C. Liu, Z-C. Lin and Y. Yu. Robust Subspace Segmentation by Low-Rank Representation. *In ICML*, 2010. 4

[11] G-C. Liu and S-C. Yan. Active Subspace: Towards Scalable Low-Rank Learning. *In Neural Computation*, 24(12):3371-3394, 2012. 4

[12] R-S. Liu, Z-C. Lin, F. Torre and Z-X. Su. Fixed-Rank Representation for Unsupervised Visual Learning. *In CVPR*, 2012. 4

[13] L. Mackey, A. Talwalkar and M. Jordan. Divide-and-Conquer Matrix Factorization. *arXiv:1107.0789*, 2011. 2, 3, 4, 5

[14] K. Nandakumar, Y. Chen, S. C. Dass and A. Jain. Likelihood Ratio-Based Biometric Score Fusion. *In TPAMI*, 30(2):342-347, 2008. 1, 2

[15] M. E. Nilsback and A. Zisserman. A Visual Vocabulary for Flower Classification. *In CVPR*, 2006. 6

[16] A. Rakotomamonjy, F. Bach, S. Canu and Y. Grandvalet. SimpleMKL, *In JMLR*,9:2491-2521, 2008. 6

[17] Z. Song, Q. Chen, Z-Y. Huang, Y. Hua and S-C. Yan. Contextualizing object detection and classification. *In CVPR*, 2011. 1

[18] O. R. Terrades, E. Valveny and S. Tabbone. Optimal Classifier Fusion in a Non-Bayesian Probabilistic Framework. *In TPAMI*, 31(9):1630-1644, 2009. 1, 2

[19] A. Vedaldi, V. Gulshan, M. Varma and A. Zisserman. Multiple Kernels for Object Detection. *In CVPR*, 2009. 1, 2

[20] C. Williams and M. Seeger. Using the Nystrom method to speed up kernel machines. *In NIPS*, 2000. 4

[21] G-N. Ye, D. Liu, I-H. Jhuo and S-F. Chang. Robust Late Fusion with Rank Minimization. *In CVPR*, 2012. 1, 2, 3, 4, 5, 6