LogDet Divergence-Based Metric Learning With Triplet Constraints and Its Applications

Jiangyuan Mei, *Student Member, IEEE*, Meizhu Liu, *Member, IEEE*,
Hamid Reza Karimi, *Senior Member, IEEE*, and Huijun Gao, *Fellow, IEEE*

I. Introduction

METRIC learning or similarity learning is a powerful tool which has been widely applied in various computer vision and pattern recognition applications, including image search [1], image annotation [2] and scene categorization [3]. The basic idea of metric learning is to study a similarity metric over the input feature space of instances. Compared with the process of feature extraction, learning an appropriate similarity metric plays a more important role in measuring the similarity of instances. We can use the same data with the same feature space to accomplish different tasks with different similarity metrics. For example, a face image set with different facial expressions can be used for either face recognition or facial expression recognition, as shown in Fig. 1. And the main difference between these two tasks lies in the selection of the similarity metric. That is to say, a good metric learning algorithm can address the difficult problem of selecting and weighting features to a certain extent.

In this paper, we use Mahalanobis distance as the similarity metric to measure the feature space of instances. The Mahalanobis distance is a standard distance metric parameterized by a Positive Semi Definite (PSD) matrix M. Given a dataset $\{x_i\}$, with $x_i \in \Re^d$, $i = 1, 2, \cdots, n$, the square Mahalanobis distance between instances x_i and x_j is defined as

$$d_M\left(x_i, x_j\right) = \left(x_i - x_j\right)^T M \left(x_i - x_j\right). \tag{1}$$

When we apply singular value decomposition to the Mahalanobis matrix, it can be decomposed as $M = H\Sigma H^T$. Here H is a unitary matrix which satisfies $HH^T = I$. Left unitary matrix is the transpose of right unitary matrix due to the symmetry of Mahalanobis matrix M. And Σ is a diagonal matrix which contains all the singular values. Thus, the square Mahalanobis distance can be rewritten as

$$\begin{aligned} d_M\left(x_i, x_j\right) &= \left(x_i - x_j\right)^T H\Sigma H^T \left(x_i - x_j\right) \\ &= \left(H^T x_i - H^T x_j\right)^T \Sigma \left(H^T x_i - H^T x_j\right) \end{aligned} \tag{2}$$

From Eqn. 2 we can see that the Mahalanobis distance has two main functions. The first one is to find the best orthogonal matrix H to remove the couplings among features and map the original feature space to a new coordinate system. The second one is to assign weights Σ to the new features according to their relationship to the task. These two functions enable Mahalanobis distance to measure the similarity among feature spaces of instances effectively in various applications. In Fig. 1, we use two different coordinate systems with weighted axial length to represent Mahalanobis metrics. The same instances with the same features can be measured and classified to different results.

In our framework, there are another two important points in our framework. The first one is the selection of constraints. In this paper, we use triplet constraints in the algorithm. From Fig. 1, we can see the main difference of learning these two Mahalanobis distance lies in the selection of triplets. The total quantity of triplets is cubic to the number of instances. Thus, we can only select part of them in metric learning process. How to make full use of the most useful triplets is one of the key problems in our algorithm. Therefore, we

Manuscript received November 27, 2013; revised May 15, 2014 and September 9, 2014; accepted September 11, 2014. Date of publication September 22, 2014; date of current version October 7, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61333012, Grant 61273201, and Grant 61203035, in part by the Deutscher Akademischer Austauschdienst Program.. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nikolaos V. Boulgouris.

J. Mei is with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150001, China (e-mail: meijiangyuan@hit.edu.cn).

H. Gao is with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150001, China, and also with the Faculty of Science King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: huijungao@gmail.com).

M. Liu is with Yahoo Labs, Yahoo Inc., New York, NY 10018 USA (e-mail: meizhu@yahoo-inc.com).

H. R. Karimi is with the Department of Engineering, Faculty of Engineering and Science, University of Agder, Kristiansand 4630, Norway (e-mail: hamid.r.karimi@uia.no).

Color versions of one or more of the figures in this paper are available online.

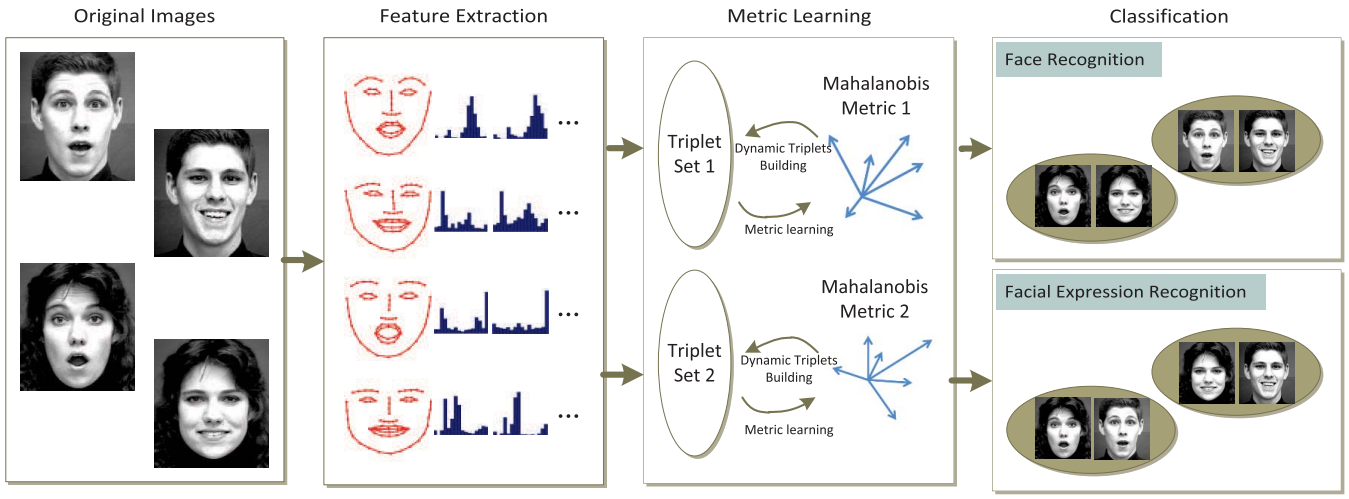


Fig. 1. The framework of our proposed LDMLT algorithm. In the original datasets, there are many face images with different facial expressions. Various fearures, such as geometric and appearance features are extracted from original images using the state-of-the-art feature extraction methods. Then, different triplet constraints are built for different tasks, such as face recognition and facial expression recognition. And the obtained Mahalanobis metric can be used in the corresponding task. In the proposed LDMLT algorithm, there is a dynamic triplets building process which can improve the performance.

propose a dynamic triplets building strategy which can update triplets using the current learned Mahalanobis distance. The feedback can further enhance the performance of the metric learning algorithm. The second point is how to deal with high dimensional feature space. The feature extraction process is sometimes aimless. In order to preserve details which might be useful in the classification task, as many features as possible would be extracted in feature extraction process. This will lead to a high dimensional feature space, which is not benefit for learning, storing and evaluating Mahalanobis matrix efficiently. Therefore, we apply a compressed representation method in our algorithm which can well solve the problem.

This paper is a comprehensive extension of our previous work [5]. The main contributions of this paper are summarized as follows. (1) We demonstrate the superiority of the triplet constraint compared with the label constraint and pairwise constraint and apply it in the online metric learning model. Meanwhile, we obtain a more accurate regularization parameter (compared with that in our previous work [5]) which can guarantee that the obtained Mahalanobis matrix is bounded between 0 and unit matrix I in the iterative process. (2) The paper applies a compressed representation method in the proposed algorithm to learn, store and evaluate the Mahalanobis matrix for high dimensional dataset efficiently. The method can reduce running time sharply at the price of losing a little classification precision. (3) This work presents a dynamic triplets building strategy which can utilize the most important triplets in metric learning according to the current Mahalanobis distance. The strategy is proven to improve the performance of metric learning significantly in the following experiments. (4) The LDMLT algorithms are applied to various applications, including facial expression recognition and image retrieval. The experimental results demonstrate the proposed algorithms can achieve improved performance.

The remainder of this paper is organized as follows. In Section II, the review of literatures is presented. Then we describe the proposed algorithms in Section III. In Section IV, experimental results on UCI database, CK+ and Caltech dataset are presented to demonstrate the effectiveness of the proposed algorithm. Finally, we draw conclusions and point out future directions in Section V.

II. Related Work

Recently, many scholars have proposed lots of metric learning algorithms to get perfect similarity metrics. Generally, these algorithms can be categorized into unsupervised and supervised metric learning [6]. In this paper, we mainly discuss the supervised metric learning which can be further categorized as offline distance metric learning and online distance metric learning.

The offline distance metric learning algorithms train Mahalanobis metric by using all the constraints at one time. The most famous one is large margin nearest neighbor (LMNN) [7] metric learning algorithm. This method applies the idea of support vector machine (SVM) to the metric learning. The objective Mahalanobis function is to maintain consistency of data in the same class while keeping a large margin at the boundaries of different categories. Another classical algorithm is the probabilistic global distance metric learning (PGDM) [8]. In this method, the metric learning is converted to a convex optimization problem with respect to their similarity relationships. In [9], the BoostMetric algorithm introduces the boosting-based technique into metric learning. The method trains trace-one rank-one matrices as weak learners, and the positive linear combination of these matrices is the objective Mahalanobis matrix. In [10], the BoostMetric algorithm is further developed. The proposed MetricBoost algorithm decomposed proximity relationships over triplets into pairwise constraints, which improves the computation efficiency to a certain degree. However, one weak point of these offline distance metric learning algorithms is the low execution efficiency, especially when the size of training samples is very large. If using the standard semidefinite program to solve the optimization problems, the computational complexity would

be $O\left(n^3\right)$, while n is the size of training samples. Meanwhile, some of these work require eigenvalue decompositions, which is also cubic to the dimensionality of the data [11].

The online distance metric learning algorithms update the Mahalanobis matrix only using one constraint at each optimization iteration. In [11], the information theoretic metric learning (ITML) algorithm minimizes the LogDet divergence between the target distance function and the predicted distance function with pairwise constraints. And the model is converted to a iteration formulation. The computational complexity of ITML is about $O\left(Nd^2\right)$, while d is the dimensionality of the data and N is the total quantity of the pair constraints. Besides, [11] also presents a online ITML algorithm, which further improve the efficiency of metric learning. The LogDet exact gradient online (LEGO) algorithm [12] improves online ITML algorithm by exactly solving the updated parameters. However, these three LogDet divergence based local distance metric learning algorithms all use pairwise constraints in the metric learning process. The pairwise constraints are strict, which will lead to conservative results.

III. LogDet Divergence Based Metric Learning With Triplet Constraints

In this section, we will describe the proposed LDMLT algorithm at length. First of all, the LogDet divergence based model with triplet constraints is presented. Then, we solve the metric learning model and obtain the updating formulation. After that, we get the regularization parameter which can maintain good properties of LDMLT algorithm. Besides, to speed up the LDMLT algorithm for high dimensional data, a compressed representation method is used to reduce the computation complexity in each iteration. Furthermore, we present a triplets building strategy to further improve the performance of metric learning.

A. Problem Formulation

As mentioned above, Mahalanobis distance function has many advantages compared with other distance functions. However, learning a Mahalanobis matrix is a complex procedure. We should pay special attention to three points in metric learning process. First, as much useful samples as possible should be trained to avoid getting in a local extremum. Second, the metric learning algorithm should be scalable with respect to the size of the training samples. Third, considering the practical application of algorithms, the labels of the training samples should be given as the most natural form. In another word, the labels should be the most easily available. The offline distance metric learning algorithms can't meet the first and second requests at the same time because they deal with all constraints at one time [12]. It often leads to a rapid growth of execution time when increasing the size of the training samples.

In this paper, we use an online metric learning model to manage the computational efficiency. Online metric learning models have two major advantages over traditional offline methods. First, in some practical applications, the system can only receive several instances or constraints at a time, and the desired Mahalanobis distance should be updated gradually over time. The offline methods can't deal with these applications in this situation. Second, some offline applications with numerous instances can be converted to online metric learning problems. Compared with offline methods, online metric learning reduces the running time dramatically since the Mahalanobis distance is optimized step by step rather than calculated at a time. Our metric learning framework is to solve the following typical online metric learning problem [11]–[13],

$$M_{t+1} = \arg\min_{M \succeq 0} D\left(M, M_t\right) + \eta_t \ell\left(M, \hat{y}_t, y_t\right), \tag{3}$$

where M_t represents the iterative Mahalanobis matrix at iteration t. And $\eta_t > 0$ is a regularization parameter which balances the regularization function $D\left(M, M_t\right)$ and loss function $\ell\left(M, \hat{y}_t, y_t\right)$. $\hat{y}_t$ and y_t respectively stands for the prediction function and the target function which will be introduced in detail in the following paragraphes.

In this framework, the first item $D\left(M, M_t\right)$ is a regularization function which is used to guarantee the stability of metric learning process. The function $D_\phi\left(\right)$ represents Bregman matrix divergence [14] between two matrices, defined as

$$\begin{aligned} D_\phi\left(M, M_t\right) = \phi\left(M\right) - \phi\left(M_t\right) \\ -tr\left(\left(\nabla\phi\left(M_t\right)\right)^T\left(M - M_t\right)\right), \end{aligned} \tag{4}$$

where the function $tr\left(\right)$ stands for the trace of a matrix. The properties of Bregman matrix divergence $D_\phi\left(M, M_t\right)$ is determined by the differentiable function $\phi\left(M\right)$. When the differentiable function $\phi\left(M\right) = -\sum_i \log\lambda_i = -\log\left(\det\left(M\right)\right)$ is chosen as the Burg entropy of the eigenvalues λ_i, the corresponding Bregman matrix divergence is called LogDet divergence [14],

$$D_{ld}\left(M, M_t\right) = tr\left(MM_t{}^{-}1\right) - \log\left(\det\left(MM_t{}^{-1}\right)\right) - n. \tag{5}$$

where n is the dimension of M.

The second item in the framework $\ell\left(M, \hat{y}_t, y_t\right)$ is the loss function measuring the loss between prediction function $\hat{y}_t$ and target function y_t at time step t. The prediction function means that the Mahalanobis distance (or the Mahalanobis distance relationship) using the Mahalanobis matrix M while the target function is the desired distance (or the desired Mahalanobis distance relationship). Obviously, when the total loss function $L_M = \sum_t \ell\left(M, \hat{y}_t, y_t\right)$ reaches its minimal, the obtained M is the closest to the desired distance function. How to build $\hat{y}_t$ and y_t is determined by the way that constructs training constraints.

There are three common constraints in learning algorithms. The first one is named as label constraint $\{x_i, w_i\}$ [15] which means the category label of instance x_i is w_i. The label constraint is not widely used in metric learning algorithms. The second one is pairwise constraint $\left\{\left(x_i, x_j\right), w_{ij}\right\}$. This constraint provides the similarity relationship information between instances x_i and x_j. $w_{ij} = 1$ indicates that x_i and x_j are similar while $w_{ij} = -1$ shows that they are dissimilar. Many famous metric learning algorithms including ITML method and LEGO all use pairwise constraints to train the

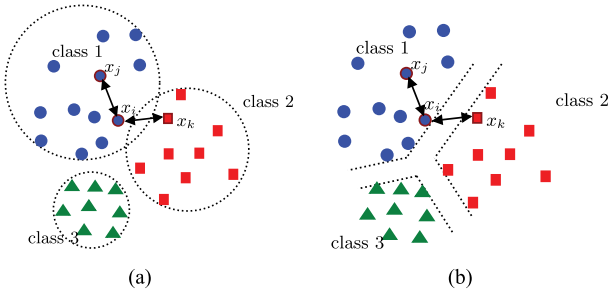


Fig. 2. Comparison of pairwise constraints and triplet constraints in metric learning. (a) The pairwise constraints require that the diameter of all hyper-spheres should smaller than $\check{c}$ while the distance between the outer boundaries of the every two hyper-spheres should be larger than $\hat{c}$. (b) The triplet constraints requires the distance between instances lying on the boundaries should be enlarged while instances in the same class should be compact.

similarity metric. They require that the obtained Mahalanobis distance M should satisfy

$$\begin{cases} d_M\left(x_i, x_j\right) < \check{c} & if \ w_{ij} = 1 \\ d_M\left(x_i, x_j\right) \geq \hat{c} & if \ w_{ij} = -1, \end{cases}$$

where $\check{c}$ and $\hat{c}$ are the desired superior limit of distance among instances in the same category and the desired lower limit of distance among instances in different categories respectively. Thus, the prediction function is $\hat{y}_t = d_M\left(x_i, x_j\right)$. The target function is $y_t = \check{c}$ if $w_{ij} = 1$; $y_t = \hat{c}$ if $w_{ij} = -1$. Although pairwise constraint is weaker than label constraint, it still has some limitations in practical applications. When using pairwise constraints as the labels of training samples, the metric learning process is to bound instances with the same class in a hyper-sphere. And the diameter of all hyper-spheres should be smaller than $\check{c}$. At the same time, the distance between the outer boundaries of every two hyper-spheres should be larger than $\hat{c}$. The obtained Mahalanobis distance functions using pairwise constraints are conservative because some of constraints are needless. For example, as shown in Fig. 2(a), these three classes have been well separated, but they can hardly meet the request that the distances among the outer boundaries of these three hyper-spheres are larger than $\hat{c}$. In order to solve this problem, literatures [4], [10] introduced a weaker representation into metric learning algorithms. This representation is called triplet constraint $\{x_i, x_j, x_k\}$, which requires that the instance x_i should be more similar to the instance x_j than instance x_k using the metric M. That is $d_M\left(x_i, x_j\right) - d_M\left(x_i, x_k\right) < -\rho$, where ρ is a desired margin between different categories. The work in [16] has demonstrated that triplet constraints can be derived from pairwise constraints, but not vice versa. Thus, triplet constraints are weaker than pairwise constraints. From Fig. 2(b) we can see, after applying the triplet constraints, metric learning is able to enlarge the distance between instances lying on the boundaries while making the instances in the same class compact.

Therefore, in the proposed framework, we use triplet constraints as the labels of the training samples. The prediction distance is $\hat{y}_t = d_M\left(x_i, x_k\right) - d_M\left(x_i, x_j\right)$ and the target distance is chosen as $y_t = \rho$. Thus, the corresponding loss function is expressed as $\ell\left(M, \hat{y}_t, y_t\right) = \max\left(0, \rho + d_M\left(x_i, x_j\right) - d_M\left(x_i, x_k\right)\right)$. When receiving a new triplet $\{x_i, x_j, x_k\}$ at time step t, if $d_{M_t}\left(x_i, x_k\right) - d_{M_t}\left(x_i, x_j\right) \geq \rho$, there is no loss using the current M_t to represent the relationship among these three instances; if $d_{M_t}\left(x_i, x_k\right) - d_{M_t}\left(x_i, x_j\right) < \rho$, the current M_t should be updated to a better Mahalanobis distance to reduce the loss. In the following section, we focus our attention on dealing with the second case.

B. Updating Formulation

The function $D_{ld}\left(M, M_t\right) + \eta_t \ell\left(M\right)$ reaches its extremum when its gradient to M is zero. The gradient of $D_{ld}\left(M, M_t\right)$ is $M_t^{-1} - M^{-1}$ while the gradient of $\eta_t \ell\left(M\right)$ is given as

$$\frac{d}{dM}\left(\eta_t \ell\left(M\right)\right) = \begin{cases} 0 & if \ \left(q_t^T M_t q_t - p_t^T M_t p_t\right) \geq \rho \\ \eta_t\left(p_t p_t^T - q_t q_t^T\right) & if \ \left(q_t^T M_t q_t - p_t^T M_t p_t\right) < \rho, \end{cases}$$

where $p_t = x_i - x_j$ and $q_t = x_i - x_k$. The second order derivative of $D_{ld}\left(M, M_t\right) + \eta_t \ell\left(M\right)$ is $M^{-2} \geq 0$ (because we require that M is a PSD in each iteration). Therefore, the extreme value is a global minimum when the gradient of $D_{ld}\left(M, M_t\right) + \eta_t \ell\left(M\right)$ to M is zero.

If $\left(q_t^T M_t q_t - p_t^T M_t p_t\right) \geq \rho$, we can get that the optimal M is equal to M_t. And the Mahalanobis distance M doesn't update in this situation. Thus, we only consider the case that $\left(q_t^T M_t q_t - p_t^T M_t p_t\right) < \rho$, and we can get the following equation:

$$M_{t+1} = \left({M_t}^{-1} + \eta_t\left(p_t p_t^T - q_t q_t^T\right)\right)^{-1}. \tag{6}$$

Matrix inverse is computationally very expensive. In order to avoid inverse, we apply the Sherman-Morrison inverse formula to solve Eqn. 6. The standard Sherman-Morrison formula is

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}.$$

However, in our updating equation, there are two items which are the outer product of vectors. To solve this problem, we assume that $\gamma_t = \left({M_t}^{-1} + \eta_t p_t p_t^T\right)^{-1}$, and Eqn. 6 is split into two standard Sherman-Morrison inverse questions,

$$\begin{cases} \gamma_t = \left({M_t}^{-1} + \eta_t p_t p_t^T\right)^{-1} \\ M_{t+1} = \left({\gamma_t}^{-1} - \eta_t q_t q_t^T\right)^{-1}. \end{cases}$$

Applying the Sherman-Morrison formula, we arrive at an analytical updating formula for M_{t+1}

$$\begin{cases} \gamma_t = M_t - \dfrac{\eta_t M_t p_t p_t^T M_t}{1 + \eta_t p_t^T M_t p_t} \\ M_{t+1} = \gamma_t + \dfrac{\eta_t \gamma_t q_t q_t^T \gamma_t}{1 - \eta_t q_t^T \gamma_t q_t}. \end{cases} \tag{7}$$

The details of the LDMLT algorithm are illustrated in Algorithm 1.

Algorithm 1 LogDet Divergence Based Metric Learning Using Triplet Constraints

Input: triplets $\{(x_i, x_j, x_k)\}$, desired margin ρ, learning rate α, total iterations N;
Output: Mahalanobis matrix M;
Initialization : Initialize $t = 0$ and $M_t{=}I$;
repeat
  $t = t + 1$; $p_t = (x_i - x_j)$; $q_t = (x_i - x_k)$;
  if $\left(q_t M_t q_t^T - p_t M_t p_t^T\right) < \rho$ **then**
    $\eta_t = \frac{\alpha}{tr\left((I-M_t)^{-1} M_t q_t q_t^T\right);}$ $\gamma_t = M_t - \frac{\eta_t M_t p_t p_t^T M_t}{1+\eta_t p_t^T M_t p_t}$;
    $M_{t+1} = \gamma_t + \frac{\eta_t \gamma_t q_t q_t^T \gamma_t}{1-\eta_t q_t^T \gamma_t q_t}$;
  end if
until $t == N$

C. Selecting η_t

In this updating equations, the regularization parameter η_t is used to control the balance of the regularization function and the loss function. On one hand, if we choose a big η_t, the M_{t+1} will be mainly updated to minimize the loss function and satisfy the target relationship among the three instances in the current triplet, which makes the metric learning process unstable. On the other hand, if η_t is too small, the M_{t+1} will have small divergence compared with the current Mahalanobis matrix M_t and every iteration will have little influence on the updating of the Mahalanobis matrix. Thus, the metric learning process will be very slow and conservative. Therefore, we should make a tradeoff between efficiency and stability when selecting η_t. Meanwhile, considering the definition of Mahalanobis metric, we also requires that the η_t should make sure that M_{t+1} is a PSD matrix in each iteration, i.e.

$$\begin{cases} \eta_t \left(p_t p_t^T - q_t q_t^T\right) + M_t^{-1} \succeq 0 \\ \eta_t \geq 0. \end{cases}$$

These are standard linear matrix inequalities (LMIs). Lots of tools can be utilized to solve the LMIs, such as LMI Solvers in MATLAB. However, using LMIs tools occupies heavy computation in each iteration. In this paper, we select a feasible solution $\eta_t = \frac{\alpha}{tr\left((I-M_t)^{-1} M_t q_t q_t^T\right)}$ in our algorithm, where α is the learning rate parameter which is chosen between 0 and 1. The following content is to demonstrate that the lower and upper bound of M_t can be restricted between zero matrix 0 and unit matrix I if using the recommended η_t. The lower bound zero matrix 0 is to guarantee that M_t is a PSD matrix at each step t. Meanwhile, the upper bound unit matrix I is to bound the total loss in the metric learning process.

Lemma 1: If $0 \preceq M_t \preceq I$, then $0 \preceq \gamma_t \preceq I$.

Proof: Firstly, let's consider the updating equation $\gamma_t = \left({M_t}^{-1} + \eta p_t p_t^T\right)^{-1}$. Since ${\gamma_t}^{-1}$ is the sum of two PSD matrices, thus $\gamma_t \succeq 0$.

Secondly, from Eqn. 7, we can get $I - \gamma_t = I - M_t + \frac{\eta_t M_t p_t p_t^T M_t}{1+\eta_t p_t^T M_t p_t}$. In this equation, since M_t is symmetric, thus $\frac{\eta_t M_t p_t p_t^T M_t}{1+\eta_t p_t^T M_t p_t} = \frac{\eta_t M_t p_t (M_t p_t)^T}{1+\eta_t p_t^T M_t p_t} \succeq 0$. Meanwhile, since $M_t \preceq I$, we can get $I - M_t \succeq 0$. Thus, we can deduce that $I - \gamma_t \succeq 0$. Hence proved.

Lemma 2: At each step t of the LDMLT algorithm,

$$\eta_t < \frac{1}{tr\left(\left(I + \left({\gamma_t}^{-1} - I\right)^{-1}\right)\gamma_t q_t q_t^T\right)} < \frac{1}{q_t^T \gamma_t q_t}.$$

Proof: In the first place, we prove the first half of the inequality $\eta_t < \frac{1}{tr\left(\left(I+\left({\gamma_t}^{-1}-I\right)^{-1}\right)\gamma_t q_t q_t^T\right)}$. The denominator can be simplified to

$$\begin{aligned} &tr\left(\left(I + \left({\gamma_t}^{-1} - I\right)^{-1}\right)\gamma_t q_t q_t^T\right) \\ &= tr\left(\left((I-\gamma_t)(I-\gamma_t)^{-1} + \gamma_t (I-\gamma_t)^{-1}\right)\gamma_t q_t q_t^T\right) \\ &= tr\left((I-\gamma_t)^{-1}\gamma_t q_t q_t^T\right). \end{aligned}$$

Since $I - \gamma_t \succeq 0$, $(I-\gamma_t)^{-1} \succeq 0$, and

$$\begin{aligned} &tr\left((I-\gamma_t)^{-1}\gamma_t q_t q_t^T\right) \\ &= tr\left((I-\gamma_t)^{-1}\left(M_t - \frac{\eta M_t p_t p_t^T M_t}{1+\eta p_t^T M_t p_t}\right) q_t q_t^T\right) \\ &< tr\left((I-\gamma_t)^{-1} M_t q_t q_t^T\right) \\ &= tr\left(\left(I - M_t + \frac{\eta M_t p_t p_t^T M_t}{1+\eta p_t^T M_t p_t}\right)^{-1} M_t q_t q_t^T\right) \\ &< tr\left((I-M_t)^{-1} M_t q_t q_t^T\right), \end{aligned}$$

thus $\eta_t = \frac{\alpha}{tr((I-M_t)^{-1} M_t q_t q_t^T)} < \frac{1}{tr\left(\left(I+\left({\gamma_t}^{-1}-I\right)^{-1}\right)\gamma_t q_t q_t^T\right)}$.

In the second place, we consider the second half of the inequality $\frac{1}{tr\left(\left(I+\left({\gamma_t}^{-1}-I\right)^{-1}\right)\gamma_t q_t q_t^T\right)} < \frac{1}{q_t^T \gamma_t q_t}$. Using Sherman Morrison Woodbury formula,

$$\left({\gamma_t}^{-1} - I\right)^{-1} = \gamma_t + \gamma_t (I-\gamma_t)^{-1}\gamma_t \succeq 0.$$

Since

$$\begin{aligned} &tr\left(\left(I + \left({\gamma_t}^{-1} - I\right)^{-1}\right)\gamma_t q_t q_t^T\right) \\ &= tr\left(\gamma_t q_t q_t^T\right) + tr\left(\left({\gamma_t}^{-1} - I\right)^{-1}\gamma_t q_t q_t^T\right) \\ &> tr\left(\gamma_t q_t q_t^T\right) = q_t^T \gamma_t q_t, \end{aligned}$$

thus $\frac{1}{tr\left(\left(I+\left({\gamma_t}^{-1}-I\right)^{-1}\right)\gamma_t q_t q_t^T\right)} < \frac{1}{q_t^T \gamma_t q_t}$. Hence proved.

Theorem 1: If $0 \preceq M_t \preceq I$, then $0 \preceq M_{t+1} \preceq I$.

Proof: At first, we consider the update equation $M_{t+1} = \gamma_t + \frac{\eta_t \gamma_t q_t q_t^T \gamma_t}{1-\eta_t q_t^T \gamma_t q_t}$. Since $\gamma_t \succeq 0$, $\eta_t \gamma_t q_t q_t^T \gamma_t = \eta_t \gamma_t q_t (\gamma_t q_t)^T > 0$. At the same time, $1 - \eta_t q_t^T \gamma_t q_t > 0$ because $\eta_t < \frac{1}{q_t^T \gamma_t q_t}$. Thus we can prove that $M_{t+1} \succeq 0$.

Then, from Eqn. 7, we get $I - M_{t+1} = I - \gamma_t - \frac{\eta_t \gamma_t q_t q_t^T \gamma_t}{1-\eta_t q_t^T \gamma_t q_t}$. The literature [17] has demonstrated that the above equation can deduce $I - M_{t+1} \succeq 0$ if $\eta_t < \frac{1}{tr\left(\left(I+\left({\gamma_t}^{-1}-I\right)^{-1}\right)\gamma_t q_t q_t^T\right)}$, as proven in Lemma 2.
Hence proved.

D. Compressed Representations for High Dimensional Datasets

In practical applications, the feature space is always with high dimensionality. We can hardly determine which features would be useful in the following classification or clustering tasks. Thus, in the feature extraction process, as much features as possible would be extracted to preserve details which might be useful. For example, in scene categorization [3], more than thousands of features are extracted. These thousands of features will occupy enormous storage space and computation time. The number of parameters involved in the Mahalanobis distance is $O\left(d^2\right)$ while the computation complexity in each iteration is more than quadratic to the dimensionality d.

As mentioned above, the Mahalanobis matrix has two main functions: removing the couplings among features and assigning weights to new features. In fact, not all original features have couplings. Thus, only a few number of off-diagonal elements in the Mahalanobis matrix make sense. We needn't to waste so much time and storage space to compute the useless elements in the Mahalanobis matrix. Therefore, we apply the compressed representations [18] method to learn, store and evaluate the Mahalanobis matrix efficiently. We constrain the objective Mahalanobis distance function M as the sum of a high-dimensional identity I^d plus a low-rank symmetric matrix M_L, expressed as

$$M = I^d + M_L = I^d + ULU^T, \tag{8}$$

where $U \in \Re^{d\times l}$ is orthogonal basis which satisfy $U^TU = I^l$, and $L \in \Re^{l\times l}$ is a symmetric matrix with $l \ll \min(n, d)$. Correspondingly, the Mahalanobis distance function at time step t can be decomposed as $M_t = I^d + UL_tU^T$. Besides, we introduce F to represent $F = I^l + L$ and F_t to stand for $F_t = I^l + L_t$. The relationship between M and F is expressed as: $M = I^d + U\left(F - I^l\right)U^T$. And learning a low dimensional symmetric matrix F is equal to learning the original Mahalanobis distance function M. The computational complexity and storage space will decrease significantly if we can obtain the updating formulation of F_t.

Theorem 2: $D_{ld}(M, M_t) = D_{ld}(F, F_t)$.

Proof: First of all, we consider the first item in Eqn. 5.

$$\begin{aligned}
&tr\left(MM_t^{-1}\right)\\
&= tr\left(\left(I^d+ULU^T\right)\left(I^d+UL_tU^T\right)^{-1}\right)\\
&= tr\left(\left(I^d+ULU^T\right)\left(I^d - U\left(I^l - \left(L_t + I^l\right)^{-1}\right)U^T\right)\right)\\
&= tr\left(I^d+ULU^T\right) - tr\left(\left(I^d+ULU^T\right)U\left(I^l - F_t^{-1}\right)U^T\right)\\
&= tr\left(F\right) + d - l - tr\left(I^l - F_t^{-1}\right) - tr\left(L\left(I^l - F_t^{-1}\right)\right)\\
&= tr\left(F\right) - tr\left(F\left(I^l - F_t^{-1}\right)\right) + d - l\\
&= tr\left(FF_t^{-1}\right) + d - l,
\end{aligned}$$

where the second equality follows from the fact of Woodbury matrix identity

$$(A + ECF)^{-1} = A^{-1} - A^{-1}E\left(C^{-1} + FA^{-1}E\right)FA^{-1},$$

and the third equality follows from the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

Then, the second item in Eqn. 5 can be converted as

$$\begin{aligned}
&\log\left(\det\left(MM_t^{-1}\right)\right)\\
&= \log\left(\det\left(\left(I^d+ULU^T\right)\left(I^d+UL_tU^T\right)^{-1}\right)\right)\\
&= \log\left(\frac{\det\left(I^d+ULU^T\right)}{\det\left(I^d+UL_tU^T\right)}\right)\\
&= \log\left(\frac{\det\left(I^l+L\right)}{\det\left(I^l+L_t\right)}\right)\\
&= \log\left(\det\left(FF_t^{-1}\right)\right),
\end{aligned}$$

where the second equality follows from the fact that $\det\left(AB^{-1}\right) = \frac{\det(A)}{\det(B)}$, and the third equality follows from the fact that $\det\left(I^m + AB\right) = \det\left(I^n + BA\right)$ for all $A \in \Re^{m\times n}$ and $B \in \Re^{n\times m}$.

Thus, we can get the following equations

$$\begin{aligned}
&D_{ld}\left(M, M_t\right)\\
&= tr\left(MM_t^{-1}\right) - \log\left(\det\left(MM_t^{-1}\right)\right) - d\\
&= tr\left(FF_t^{-1}\right) + d - l + \log\left(\det\left(FF_t^{-1}\right)\right) - d\\
&= tr\left(FF_t^{-1}\right) + \log\left(\det\left(FF_t^{-1}\right)\right) - l = D_{ld}\left(F, F_t\right).
\end{aligned}$$

Hence proved.

Assume that $\bar{X} = U^TX$ represents the reduced-dimensional data under the orthogonal basis U, then we can get the corresponding variables $\bar{p}_t = U^T\left(x_i^t - x_j^t\right) = U^Tp_t$ and $\bar{q}_t = U^T\left(x_i^t - x_k^t\right) = U^Tq_t$. Thus, the loss function can be rewritten as $\ell(F) = \bar{p}_t^TF\bar{p}_t - \bar{q}_t^TF\bar{q}_t + \bar{\rho}$, where $\bar{\rho} = p_t^Tp_t - \bar{p}_t^T\bar{p}_t - q_t^Tq_t + \bar{q}_t^T\bar{q}_t + \rho$. Therefore if $\left(\bar{q}_t^TF_t\bar{q}_t - \bar{p}_t^TF_t\bar{p}_t\right) \geq \bar{\rho}$, the optimal F is equal to the current F_t. And the Mahalanobis distance F doesn't update in this situation. If $\left(\bar{q}_t^TF_t\bar{q}_t - \bar{p}_t^TF_t\bar{p}_t\right) < \bar{\rho}$, the updating formulation of F_t is expressed as

$$\begin{cases}
\Gamma_t = F_t - \dfrac{\bar{\eta}_tF_t\bar{p}_t\bar{p}_t^TF_t}{1 + \bar{\eta}_t\bar{p}_t^TF_t\bar{p}_t}\\
F_{t+1} = \Gamma_t + \dfrac{\bar{\eta}_t\Gamma_t\bar{q}_t\bar{q}_t^T\Gamma_t}{1 - \bar{\eta}_t\bar{q}_t^T\Gamma_t\bar{q}_t},
\end{cases} \tag{9}$$

where the parameter $\bar{\eta}_t$ is set as $\bar{\eta}_t = \frac{\alpha}{tr\left(\left(I^k - F_t\right)^{-1}F_t\bar{q}_t\bar{q}_t^T\right)}$. Using the compressed representations technique, the computational complexity reduces from $O\left(d^2\right)$ to $O\left(l^2\right)$ per iteration. At the same time, the storage space in the metric learning process also reduces sharply. The corresponding algorithm is named as LDMLT_CR, and the details are presented in Algorithm 2.

How to build an appropriate orthogonal basis U is a key step in this LDMLT_CR algorithm. In [19], the authors provide several practical methods to build the basis. If the algorithm is applied on offline applications, one efficient way is to choose the first l left singular vector to build U after applying the singular value decomposition (SVD) to the original training data X. However, one weak point of this method is that it can not deal with online applications. The reason is that the instances of online applications can't be obtained at the orthogonal basis building step, which is regarded as

Algorithm 2 LDMLT_CR for High Dimensional Datasets

Input: training samples $X = \{x_i\}$, triplets $\{(x_i, x_j, x_k)\}$, desired margin ρ, learning rate α, total iterations N, orthogonal basis U;
Output: Mahalanobis matrix M;
Initialization : Initialize $t = 0$, F_t=I;
repeat
$t = t + 1$;
$p_t = (x_i - x_j)$; $q_t = (x_i - x_k)$; $\bar{p}_t = U^T p_t$; $\bar{q}_t = U^T q_t$;
$\bar{\rho} = p_t^T p_t - \bar{p}_t^T \bar{p}_t - q_t^T q_t + \bar{q}_t^T \bar{q}_t + \rho$;
if $\bar{q}_t^T F \bar{q}_t - \bar{p}_t^T F \bar{p}_t < \bar{\rho}$ **then**
$\bar{\eta}_t = \frac{\alpha}{tr\left((I^l - F_t)^{-1} F_t \bar{q}_t \bar{q}_t^T\right)}$; $\Gamma_t = F_t - \frac{\bar{\eta}_t F_t \bar{p}_t \bar{p}_t^T F_t}{1 + \bar{\eta}_t \bar{p}_t^T F_t \bar{p}_t}$;
$F_{t+1} = \Gamma_t + \frac{\bar{\eta}_t \Gamma_t \bar{q}_t \bar{q}_t^T \Gamma_t}{1 - \bar{\eta}_t \bar{q}_t^T \Gamma_t \bar{q}_t}$;
end if
until $t == N$
$M_t = I^d + U\left(F_t - I^l\right)U^T$.

a preprocess of the proposed method. At the same time, the computational complexity of traditional SVD algorithm is about $O\left(n^2 d + n d^2 + d^3\right)$ when $d \ll n$. Therefore, a good choice is to use the a truncated incremental SVD [20] to obtain the orthogonal basis U when applying on online applications. The computational complexity of truncated incremental SVD is about $O\left(nl^3\right)$ when $l \ll d$. The truncated incremental SVD can sharply reduce the computation time at the price of losing a little precision.

E. Dynamic Triplets Building Strategy

When training a Mahalanobis matrix for practical applications, it is impossible to utilize all triplets because the total quantity of triplets is cubic to the volume of training data. We need to select the most useful triplets and remove redundant ones. The traditional metric learning algorithms including ITML choose the pairwise constraints randomly. In [4], a practical way to build triplet constraints is proposed. The triplets which are at the boundaries of different categories are selected. In detail, for each x_i, instances $\{x_j\}$ which has the largest Euclidean distance in the same category and instances $\{x_k\}$ which has the nearest Euclidean distance in the different category are selected, and these $\{(i, j, k)\}$ are used to build the training triplets. However, these previous works all regard building triplet (pairwise) constraints as a preprocess step. Once built, these constraints are not updated in the process of metric learning. In fact, these constraints are selected randomly or based on the Euclidean distance. With the updating of the Mahalanobis matrix, some triplets do not play critical roles while some unselected triplets become important.

We propose a novel triplets building strategy in this section. The metric learning process is broken into several cycles. First of all, the triplets at the boundaries of different categories are chosen in the initial cycle, which is the same as that in [4]. Then, we obtain the Mahalanobis matrix M_t using the proposed LDMLT (or LDMLT_CR) algorithm after a metric learning cycle. We use this M_t to measure

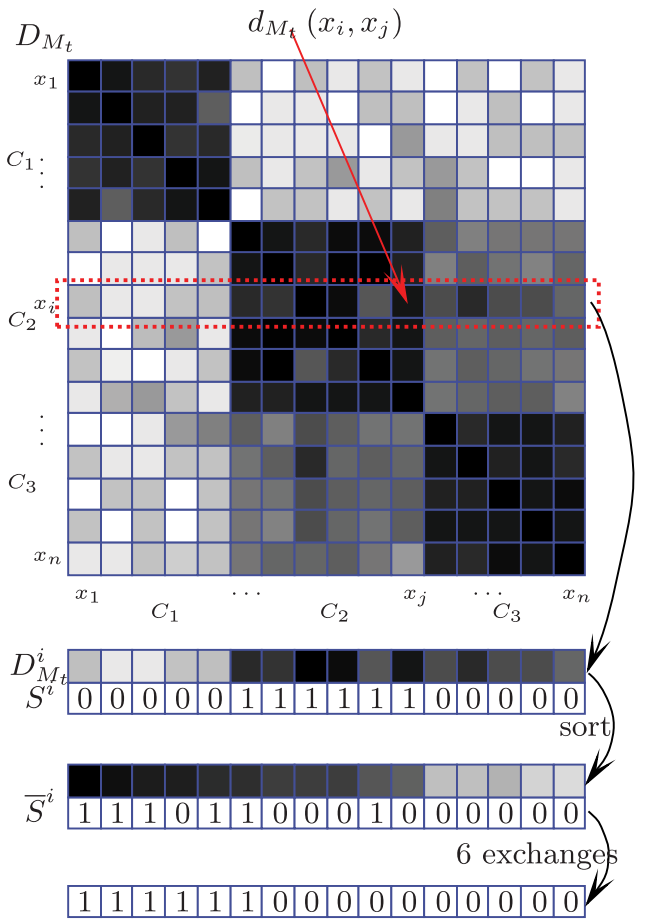


Fig. 3. The framework of our proposed triplets building strategy.

the performance on the training data X. After that, new triplets in the next cycle are selected based on the measurement.

The key problem in this method is how to measure the performance of M_t on the training dataset X. Two matrices are defined in this section. The first one is Mahalanobis distance matrix $D_{M_t} = \left\{d_{M_t}\left(x_i, x_j\right)\right\}$, which represents the Mahalanobis distance of all the sample pairs $\{(i, j)\}$ in X. The second one is the similar matrix $S = \left\{s\left(x_i, x_j\right)\right\}$, which gives the relationship information of sample pairs. If x_i and x_j belongs to the same category, $s\left(x_i, x_j\right) = 1$, otherwise, $s\left(x_i, x_j\right) = 0$.

For example, we have instances x_i, $i = 1, 2, \cdots, n$ with 3 categories C_1, C_2 and C_3. The instances in X have been sorted according to their categories, and the D_{M_t} is shown as the matrix in Fig. 3. The deep color means the value of the element is small while light color represents a large value. The ideal situation is that the Mahalanobis distances among instances in the same categories should be all smaller than that in different categories. In another word, the color in the diagonal blocks (intra-blocks) should be deeper than the color in other blocks (inter-blocks). However, the color of blocks between Class C_2 and Class C_3 is deeper than other inter-blocks, and similar to the intra-block of Class C_2. That means the obtained Mahalanobis distance M_t don't have good performance on distinguishing Class C_2 and Class C_3.

Algorithm 3 LDMLT (or LDMLT_CR) Algorithm With Dynamic Triplets Building Strategy

Input: training samples $X = \{x_i\}$, desired margin ρ, learning rate α, $S = \{s(x_i, x_j)\}$;
Output: Mahalanobis matrix M;
Initialization : Initialize $cycle = 0$, $t = 0$, M_t=I;
repeat
  $cycle = cycle + 1$; $D_{M_t} = \{d_{M_t}(x_i, x_j)\}$
  Compute the disorder Θ_i and the total disorder τ;
  Select $\frac{\Theta_i}{\sum_i \Theta_i} N$ triplets which use x_i, $i = 1, 2, ..., n$;
  All the triplets are used to build a new set $\{(x_i, x_j, x_k)\}$;
  Update the M_t using the Algorithm 1 (or Algorithm 2);
until τ converge

More constraints should be selected from triplets which use instances in these two categories.

We extract the i^{th} row in D_{M_t} which is expressed as $D^i_{M_t} = \{d_M(x_i, x_j)\}$ and the corresponding $S^i = \{s(x_i, x_j)\}$, $j = 1, 2, \cdots, n$. If we sort the vector $D^i_{M_t}$ in ascending order, we can see the corresponding S^i is also changed to $\overline{S}^i$. The ideal case is that the $\overline{S}^i$ should be in a descending order. However, the fact is that some 0 in the $\overline{S}^i$ is in front of 1, which means that the Mahalanobis distance between some dissimilar pairs is smaller than that of similar pairs using the current M_t. If we want to reorder the elements in the $\overline{S}^i$, we should exchange the disordered 0 and 1 elements 6 times. And we named this exchange count as the disorder Θ_i. We use the normalized disorder $\theta_i = \Theta_i / \sum_i \Theta_i$ to determine the proportion of triplets which contain x_i. When building triplets including x_i, the method is similar to the method in [4]. The main difference is that we use the current M_t instead of Euclidean distance. It worth noting that we can also use total disorder $\tau = \sum_i \Theta_i$ to judge if the algorithm is converged. The details of the LDMLT (or LDMLT_CR) algorithm with dynamic triplets building strategy are illustrated in Algorithm 3.

The proposed dynamic triplets building strategy provides a feedback from the current obtained Mahalanobis matrix to triplet constraints. This is one main contribution of our paper. Most traditional metric learning methods do nothing about the constraints selection when the learning process begins. Therefore, the Mahalanobis matrices obtained from these methods are not stable because the randomly chosen constraints will lead to different learning results. Our method chooses the most useful triplets in each cycle, and the Mahalanobis matrix is gradually updated to the ideal Mahalanobis matrix after several cycles. However, one weak point of the proposed strategy is the low computation efficient. The main time consumption is the calculation of D_{M_t}, which is quadratic to the number of instances. Meanwhile, this strategy can't be used in online applications by now.

IV. Experiments and Results

In this section, we firstly conduct experiments to compare the classification performance of the proposed method with the state-of-the-art metric learning methods on the UCI machine learning repository.[1] Then, we apply the LDMLT algorithm on facial expression recognition. The experiment is to validate the dynamic triplets building strategy. After that, we evaluate the LDMLT_CR method on object recognition and image retrieval tasks on Caltech datasets [26]. The experiments are to demonstrate the improved performance of the proposed algorithms on real applications compared with the state-of-the-art methods. All experiments are tested in MATLAB 2012a, and all tests are implemented on a computer with Intel(R) Core(TM) i5-2400, 3.10GHz CPU, 4G RAM, and Windows 7 64-bit operating system. The code of our LDMLT and LDMLT_CR algorithms can be downloaded from the website.[2]

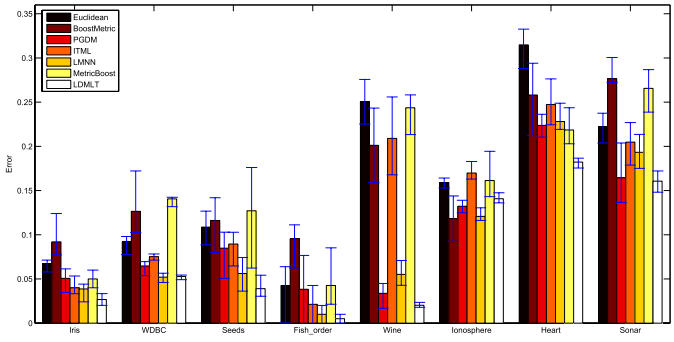
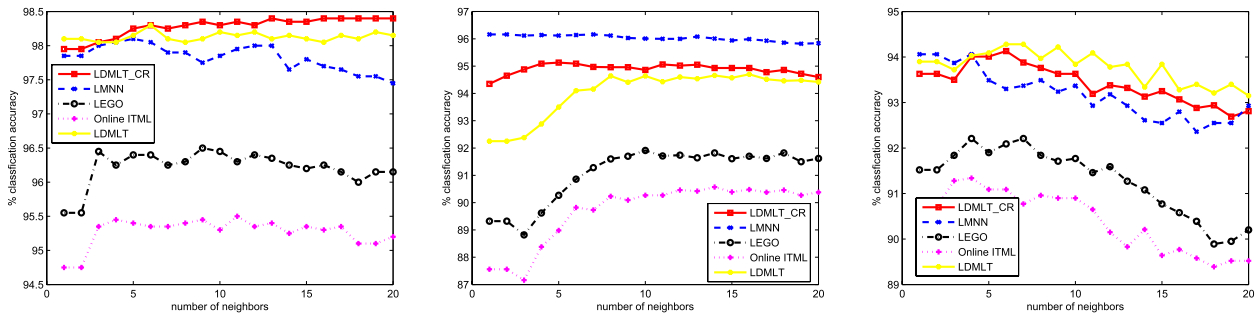


Fig. 4. Classification error rates of different metric learning methods on 8 selected datasets in UCI repository.

A. Experiments on the UCI Database

In the first experiment, we'd like to compare our LDMLT algorithm with some basic classification methods and the state-of-the-art metric learning algorithms, including Euclidean distance, BoostMetric, PGDM, ITML, LMNN and MetricBoost. The test data are eight low dimensionality datasets selected from the UCI machine learning repository, including "Iris", "WDBC", "Seeds", "Fish_order", "Wine", "Ionosphere", "Sonar", and "Heart". The performance is evaluated according to the classification error rates using the K-Nearest Neighbor (K-NN) classification, the variable K in K-NN classification is chosen as 5. In this experiment, all classification results are obtained using 5-fold cross validation and final statistical results are evaluated after 5 runs. In our LDMLT algorithm, the parameter ρ is set as the difference between the 90^{th} and 10^{th} percentiles of the distribution of distances between sample pairs in the training data. The dynamic triplets building strategy is used to strengthen the performance of LDMLT algorithm. And the cycle of dynamic triplets building process is set as 10. In each cycle, the quantity of triplets is $N = 5n$, where n is the number of training instances. The parameter α is set as $\alpha = 2/N$. The results are summarized in Fig. 4. The error bars reveal the average classification error rates. The maximum and minimum error rates in 5 runs are also shown in Fig. 4.

[1] http://archive.ics.uci.edu/ml/
[2] http://www.mathworks.com/matlabcentral/fileexchange/46437-ldmlt-zip



(a) (b) (c)

Fig. 5. The comparison of classification precision of metric learning algorithms on 3 selected high dimensional datasets in UCI repository. (a) Experimental results on dataset "Multiple Features"; (b) Experimental results on dataset "ISOLET"; (c) Experimental results on dataset "Semeion Handwritten Digit".

From the experimental results we can see, the proposed LDMLT algorithm achieves the best accuracy for all datasets except "Ionosphere". Besides, the distribution of error rates of LDMLT algorithm is the most concentric, which indicates that LDMLT is more robust than other methods. The reasons for the good performance can be explained as follows. On one hand, in LMNN, BoostMetric and MetricBoost algorithms, a gradual loss of precision is inevitable in eigenvalue computations and semi-definite programming. However, the proposed objective function is more intuitive, and the model is much simpler. Thus, the performance of LDMLT is much better than LMNN, BoostMetric and MetricBoost. On the other hand, the PDGM and ITML methods train Mahalanobis matrix using pairwise constraints while the proposed algorithm utilizes triplets as the labels of the training samples. As mentioned above, pairwise constraints are stronger than triplet constraints, so PDGM and ITML would get conservative results in some situation while the proposed method can achieve more accurate results. Moreover, the dynamic triplets building strategy helps the proposed method further improve the precision and robustness of classification. Thus, the proposed method achieves the best performance compared with these methods. It deserves noting that our method can't get good classification results on dataset "Ionosphere". This dataset provides the radar data collected by a system in Goose Bay, Labrador. And instances in this dataset are labelled as "Good" radar and "Bad" radar. The difference among "Bad" instances is large than the difference between "Good" instances and "Bad" instances. In this situation, the proposed dynamic triplets building strategy can't work well. One of solutions is to combine the proposed method with kernel methods.

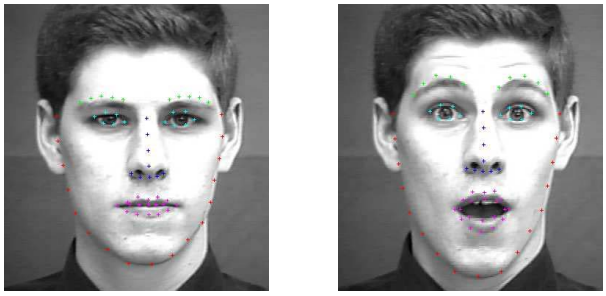
Then, another experiment is to demonstrate the performance of the proposed LDMLT_CR algorithm on high dimensional datasets. In this experiment, we select three high dimensional datasets from the UCI machine learning repository, i.e. "Multiple Features", "ISOLET" and "Semeion Handwritten Digit". These three datasets are with more than hundreds of attributes. In our algorithm, we use a 30 dimension compressed matrix to represent the Mahalanobis matrix. The parameter $\bar{\rho}$ is simply set as 0. In order to get a high classification accuracy, the dynamic triplets building strategy is also applied in the LDMLT_CR algorithm and the total cycle is $10 \sim 20$.

TABLE I

THE COMPARISON OF RUNNING TIME OF METRIC LEARNING METHODS ON 3 SELECTED HIGH DIMENSIONAL DATASETS IN UCI REPOSITORY

	Multiple Features	ISOLET	Semeion Digit
Instances	2600	7797	1593
Attributes	649	617	256
Classes	10	26	10
LDMLT	642s	4279s	56s
online ITML	132s	495s	15s
LEGO	128s	482s	**14s**
fast LMNN	195s	729s	18s
LDMLT_CR	**18s**	**142s**	15s

In each cycle, the quantity of triplets N is set as $N = 5n$. The parameter α is set as $\alpha = 2/N$. We compare our method with LDMLT algorithm as well as 3 other state-of-the-art fast metric learning algorithms, including fast LMNN, LEGO and online ITML. Fig. 5 shows the comparison of classification accuracy on these three datasets. The results reveals that the proposed method have similar accuracy to the LDMLT and fast LMNN but outperforms online ITML and LEGO. Besides, our method has a better computational efficiency than other four methods, which is shown in Table I. In this comparison, the total iterations of LDMLT, LEGO and online ITML is $5n$, which is the same as the iterations of LDMLT_CR in one cycle. However, the computational efficiency of LDMLT, LEGO and online ITML is very low when the dimensionality of data d is high. The computational complexity of LEGO and online ITML is about $O\left(Nd^2\right)$. Although the fast LMNN algorithm also reduces the dimensionality of data to 30 before metric learning process, the algorithm also requires heavy running time. The reason is that there are high dimensional semidefinite program in the algorithm. In our algorithm, the main time is spent on the dynamic triplet building process. Therefore, our method has the best performance when the dataset is with high dimension and small amount of instances, such as the dataset "Multiple Features". Compared with LDMLT method, LDMLT_CR reduces lots of computation time but gets the similar classification performance of high dimension data. One of reasons is that the LDMLT_CR can run more iterations in less running time than LDMLT method, which will help improve the classification precision.



(a) (b)

Fig. 6. The position of 68 landmarks change when facial expression starts from neutral pose to peak formation. (a) landmarks on the beginning formation of the expression; (b) landmarks on the peak formation of the expression.

TABLE II
CONFUSION MATRIX OF FACIAL EXPRESSION CLASSIFICATION USING LDMLT WITH RANDOM TRIPLETS

	An	Co	Di	Fe	Ha	Sa	Su
An	**77.78**	8.44	4.00	0	0	9.78	0
Co	27.78	**43.33**	10.00	0	6.67	11.11	1.11
Di	4.75	1.69	**82.71**	0.34	10.17	0.34	0
Fe	5.60	3.20	3.20	**49.60**	21.60	14.40	2.40
Ha	0.58	2.32	4.35	0.58	**90.72**	0	1.45
Sa	20.71	7.14	1.43	4.29	5.00	**61.43**	0
Su	0	1.20	0	0.24	2.17	0	**96.39**

B. Experiments on the Extended Cohn-Kanade Dataset

Next we apply our method on facial expression recognition. The database of facial expression is selected as the Extended Cohn-Kanade Dataset (CK+) [21]. This dataset contains 593 sequences from 123 subjects, 327 of which are labeled as one of seven typical expressions. Each sequence consists of 10 to 60 frames. The first frames of sequences are neutral and the last ones are the peak formation of the expressions. At the same time, 68 point landmarks for each image are also provided along with the dataset, which makes it very suitable for testing facial expression recognition algorithms. These landmarks are located around the outlines of face, brows, nose, eyes and mouse, as shown in Fig. 6.

In our experiment, we only use the last frame in each facial expression sequence for facial expression recognition. We extract 18 geometric features using 68 points landmarks from these facial expression images. These features are similar to the features of the frontal-view model in [25], recording the relative position of facial components, such as the extent that mouth opens. In the following experiment, leave one subject out cross validation method was used in the whole scenario.

Table. II gives the confusion matrix of facial expression classification using LDMLT with random triplets. From the results, we can see the obtained Mahalanobis matrix perform well on happiness and surprise. The reason is that the features extracted from these expressions are easy to discriminate from other expressions. However, when it comes to other five facial expressions, they are easy to be confused as their distinguishable characteristics lies in different features. For example, contempt is very easy to confused with anger, because their main difference is the curvature of eyebrows. However, this is not an important feature when compared with other features and it will not be emphasized in the obtained Mahalanobis matrix. If the triplets are selected randomly, this feature may not be focused on to make a distinction between contempt and anger. When applying the dynamic triplets building strategy, this problem can be well solved. Although the Mahalanobis matrix obtained in the first cycle may not distinguish contempt and anger very well, we will pay special attention on the triplets relating to these two facial expressions in next cycles. And the recognition accuracy between contempt and anger will improve. Table. III shows the confusion matrix of facial expression classification using LDMLT with dynamic triplets building strategy. Happiness and surprise all keep good performance while the recognition accuracy of other facial expressions increase significantly.

TABLE III
CONFUSION MATRIX OF FACIAL EXPRESSION CLASSIFICATION USING LDMLT WITH DYNAMIC TRIPLETS BUILDING STRATEGY

	An	Co	Di	Fe	Ha	Sa	Su
An	**84.00**	3.56	2.22	0.89	0.44	8.89	0
Co	15.56	**72.22**	2.22	0	0	8.89	1.11
Di	2.03	0.68	**92.88**	0	3.05	1.02	0.34
Fe	8.00	6.40	4.80	**64.80**	7.20	5.60	3.20
Ha	0.29	0.58	0	0	**99.13**	0	0
Sa	9.29	6.43	2.86	3.57	2.14	**75.71**	0
Su	0.48	0.96	1.20	1.45	0.24	0.48	**95.18**

TABLE IV
CLASSIFICATION ACCURACY OF THE STATE-OF-THE-ART FACIAL EXPRESSION RECOGNITION ALGORITHMS ON CK+

References	[21]	[22]	[23]	[24]	Proposed
Anger	75.0	70.1	81.8	**95.6**	84.0
Contempt	**84.4**	52.4	55.6	82.1	72.2
Disgust	94.7	92.5	85.7	**96.6**	92.9
Fear	65.2	**72.1**	40.0	58.3	64.8
Happy	**100.0**	94.2	98.4	97.1	99.1
Sadness	68.0	45.9	48.1	58.6	**75.7**
Surprise	96.0	93.6	**98.8**	97.6	95.2
Average	88.3	82.3	82.6	**89.9**	88.8

Then, we compare the performance of our algorithm with the some state-of-the-art facial expression recognition approaches, including active appearance models (AAM) tracked similarity-normalized shape (SPTS) and canonical appearance (CAPP) features with linear SVM [21], constrained local models (CLM) tracked SPTS and SAPP features with linear SVM [22], emotion avatar image (EAI) and local phase quantization (LPQ) features with linear SVM [23] and weighted component-based feature descriptor algorithm [24]. Table. IV gives the classification accuracy of the state-of-the-art facial expression recognition algorithms and the proposed method. Our method only use features extracted from a single frame while other methods all use dynamic features. However, the performance of the proposed method is better than methods in [21]–[23], ranking the second among all methods. An interesting phenomenon is that the classification accuracy of each categories of the proposed method is more balanced than other methods. The reason is that the dynamic triplets building strategy can focus on the categorizes with bad classification performance in each iteration, which will improve the whole performance of facial expression recognition.

TABLE V
MEAN AVERAGE PRECISION, PRECISION AT TOP 1, 10, AND 50 AND RUNNING TIME OF ALL COMPARED METHODS ON CALTECH 256 DATASET. VALUES ARE AVERAGES OVER 5 CROSS VALIDATION FOLDS. ± VALUES ARE THE STANDARD DEVIATION ACROSS THE 5 FOLDS

10 classes	LDMLT_CR	OMLLR	OASIS	LEGO	LMNN
Avg. prec	**51**$_{\pm 1.6}$	41$_{\pm 1.6}$	33$_{\pm 1.6}$	27$_{\pm 0.8}$	24$_{\pm 1.6}$
Top 1 prec	**65**$_{\pm 1.0}$	51$_{\pm 2.8}$	43$_{\pm 4.0}$	39$_{\pm 4.8}$	38$_{\pm 5.4}$
Top 10 prec	**56**$_{\pm 2.0}$	45$_{\pm 2.2}$	38$_{\pm 1.3}$	32$_{\pm 1.2}$	29$_{\pm 2.1}$
Top 50 prec	**42**$_{\pm 1.2}$	34$_{\pm 1.0}$	23$_{\pm 1.5}$	20$_{\pm 0.5}$	18$_{\pm 1.5}$
running time	88$_{\pm 12}$	342$_{\pm 31}$	**42**$_{\pm 15}$	143$_{\pm 44}$	337$_{\pm 169}$
20 classes	LDMLT_CR	OMLLR	OASIS	LEGO	LMNN
Avg. prec	**32**$_{\pm 1.7}$	23$_{\pm 1.3}$	12$_{\pm 0.4}$	21$_{\pm 1.4}$	16$_{\pm 1.2}$
Top 1 prec	**48**$_{\pm 1.1}$	33$_{\pm 1.7}$	21$_{\pm 1.6}$	29$_{\pm 2.6}$	26$_{\pm 2.7}$
Top 10 prec	**38**$_{\pm 1.9}$	26$_{\pm 1.6}$	16$_{\pm 0.4}$	24$_{\pm 1.9}$	20$_{\pm 1.4}$
Top 50 prec	**25**$_{\pm 1.7}$	20$_{\pm 1.0}$	10$_{\pm 0.3}$	15$_{\pm 0.4}$	13$_{\pm 0.6}$
running time	297$_{\pm 29}$	550$_{\pm 43}$	**45**$_{\pm 8}$	14$_{\pm 0.7}$	533$_{\pm 49}$
50 classes	LDMLT_CR	OMLLR	OASIS	LEGO	LMNN
Avg. prec	**17**$_{\pm 0.4}$	14$_{\pm 0.3}$	12$_{\pm 0.4}$	9$_{\pm 0.4}$	8$_{\pm 0.4}$
Top 1 prec	**28**$_{\pm 0.9}$	22$_{\pm 1.4}$	21$_{\pm 0.4}$	18$_{\pm 0.7}$	18$_{\pm 1.3}$
Top 10 prec	**20**$_{\pm 0.5}$	17$_{\pm 0.3}$	16$_{\pm 0.4}$	13$_{\pm 0.6}$	12$_{\pm 0.5}$
Top 50 prec	**12**$_{\pm 0.4}$	**12**$_{\pm 0.4}$	10$_{\pm 0.4}$	8$_{\pm 0.3}$	7$_{\pm 0.2}$
running time	1632$_{\pm 103}$	731$_{\pm 28}$	**25**$_{\pm 2}$	711$_{\pm 28}$	960$_{\pm 80}$

C. Experiments on Caltech 256

In this part, the proposed metric learning algorithm is applied on image retrieval tasks. The experiments are conducted on Caltech-256 object category datasets [26], which consist of pictures of objects from 256 categories. In our experiments, we use VLFeat [27] to extract pyramid histogram of visual words (PHOW) features [28] of the images in Caltech datasets. We randomly select 20 images from each category to construct a visual word dictionary with 300 words. After several processes (please refer [27] and the caltech-101-code[3] for details), we obtain the PHOW features with 3600 dimensions.

Our experiment follows the process presented in literatures [3] and [29]. 40 training images and 25 test images are randomly selected from each category. The number of categories respectively set as 10, 20 and 50 (please refer [3] and [29] for the details of these 10 classes, 20 classes and 50 classes subsets). The performance index is selected as ranking precision measures, including top k (e.g. 1, 10 and 50) precision and the average precision [3] which is based on nearest neighbors. The total iteration is also set as $35K$, as same as that in [3] and [29]. The proposed method is compared with some state-of-the-art methods, including online metric learning via low rank (OMLLR) [3], online algorithm for scalable image similarity learning (OASIS) [29], LEGO and LMNN. In this experiment, we use 3600 dimensional PHOW features in our method which is different with other four methods. These four methods use edge and color histograms to represent images. Statistical results after 5 cross validation folds are presented in Table V. The results of other four methods are reported by literatures [3] and [29]. Our method outperforms all other four methods in terms of classification precision. One possible reason is that the proposed LDMLT_CR method can deal with high dimensional data, which allows more features for classification. In fact, if using our 3600 dimensional PHOW features, the LEGO and LMNN can't even work. However, compared with OASIS method, the computational efficiency of the LDMLT_CR is very low, and it increases fast when the number of category increases. The main reason is that the running time of the dynamic triplets building is quadratic to the quantity of training instances.

[3]http://www.vlfeat.org/applications/caltech-101-code.html

V. CONCLUSION

Feature selection and weighting have always been a difficult problem in practical applications. This paper illustrates a novel image processing and pattern recognition framework which is based on LDMLT (or LDMLT_CR) algorithm. This framework can solve the problem to a certain extent. In this framework, the same instances with the same features can be applied in different applications with different Mahalanobis metrics. In this paper, a LDMLT algorithm which is based on LogDet divergence and triplet constraints is presented. With the designed regularization parameter η_t, the LDMLT algorithm can learn the Mahalanobis distance metric accurately and efficiently. Besides, a compressed representation method applied in our LDMLT algorithm, and the so called LDMLT_CR algorithm can learn, store and evaluate high dimensional Mahalanobis matrix efficiently. Furthermore, to further improve the classification precision of the learned Mahalanobis metrics, we use a novel dynamic triplets building strategy to strengthen LDMLT and LDMLT_CR algorithms. Experiments on various benchmark datasets and comparisons with other state-of-the-art methods demonstrate the robustness, high precision and efficiency of our algorithm. One drawback of the proposed framework is the relatively low efficiency of the dynamic triplets building strategy. Therefore, in our further research, we will focus on solving this problem.

REFERENCES

[1] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[2] B. Geng, D. Tao, and C. Xu, "DAML: Domain adaptation metric learning," *IEEE Trans. Image Process.*, vol. 20, no. 10, pp. 2980–2989, Oct. 2011.

[3] Y. Cong, J. Liu, J. Yuan, and J. Luo, "Self-supervised online metric learning with low rank constraint for scene categorization," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3179–3191, Aug. 2013.

[4] M. Liu and B. C. Vemuri, "A robust and efficient doubly regularized metric learning approach," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 646–659.

[5] J. Mei, M. Liu, H. R. Karimi, and H. Gao, "LogDet divergence based metric learning using triplet labels," in *Proc. ICML Workshop Divergences Divergence Learn.*, 2013, pp. 1–9.

[6] L. Yang and R. Jin, *Distance Metric Learning: A Comprehensive Survey*, vol. 2. Lansing, MI, USA: Michigan State Univ., 2006.

[7] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, vol. 18. Cambridge, MA, USA: MIT Press, 2006, p. 1473.

[8] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," *Advances in Neural Information Processing Systems*, vol. 15. Cambridge, MA, USA: MIT Press, 2002, pp. 505–512.

[9] C. Shen, J. Kim, L. Wang, and A. van den Hengel, "Positive semidefinite metric learning with boosting," in *Advances in Neural Information Processing Systems*, vol. 22. Red Hook, NY, USA: Curran Associates, 2009, pp. 629–633.

[10] J. Bi, D. Wu, L. Lu, M. Liu, Y. Tao, and M. Wolf, "AdaBoost on low-rank PSD matrices for metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2617–2624.

[11] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 209–216.

[12] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Advances in Neural Information Processing Systems 21*. Red Hook, NY, USA: Curran Associates, 2008, pp. 761–768.

[13] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 94.

[14] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 505–512.

[15] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, May 2007.

[16] Q. Wang, P. C. Yuen, and G. Feng, "Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions," *Pattern Recognit.*, vol. 46, no. 9, pp. 2576–2587, 2013.

[17] P. Jain, B. Kulis, and I. Dhillon, "Online linear regression using burg entropy," Dept. Comput. Sci., Univ. Texas at Austin, Austin, TX, USA, Tech. Rep. TR-07-08, 2007.

[18] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon, "Metric and kernel learning using a linear transformation," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 519–547, 2012.

[19] J. V. Davis and I. S. Dhillon, "Structured metric learning for high dimensional problems," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 195–203.

[20] H. Zhao, P. C. Yuen, and J. T. Kwok, "A novel incremental principal component analysis and its application for face recognition," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 4, pp. 873–886, Aug. 2006.

[21] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn–Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 94–101.

[22] S. W. Chew, P. Lucey, S. Lucey, J. Saragih, J. F. Cohn, and S. Sridharan, "Person-independent facial expression detection using constrained local models," in *Proc. IEEE Int. Conf. Automat. Face Gesture Recognit. Workshops*, Mar. 2011, pp. 915–920.

[23] S. Yang and B. Bhanu, "Understanding discrete facial expressions in video using an emotion avatar image," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 4, pp. 980–992, Aug. 2012.

[24] X. Huang, G. Zhao, M. Pietikäinen, and W. Zheng, "Expression recognition in videos using a weighted component-based feature descriptor," in *Proc. 17th Scandin. Conf. Image Anal.*, 2011, pp. 569–578.

[25] M. Pantic and L. J. M. Rothkrantz, "Expert system for automatic analysis of facial expressions," *Image Vis. Comput.*, vol. 18, no. 11, pp. 881–905, 2000.

[26] G. Griffin, A. Holub, and P. Perona, *Caltech-256 Object Category Dataset*. Pasadena, CA, USA: California Institute Technology, 2007.

[27] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proc. Int. Conf. Multimedia*, 2010, pp. 1469–1472.

[28] A. Bosch, A. Zisserman, and X. Muñoz, "Image classification using random forests and ferns," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[29] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, Mar. 2010.