Large Margin Local Metric Learning

Julien Bohné[1,2], Yiming Ying[3], Stéphane Gentric[2], and Massimiliano Pontil[1]

[1] University College London, Department of Computer Science, London, UK
m.pontil@cs.ucl.ac.uk
[2] Safran Morpho, Issy-les-Moulineaux, France
{julien.bohne,stephane.gentric}@morpho.com
[3] University of Exceter, Department of Computer Science, Exceter, UK
y.ying@exeter.ac.uk

Abstract. Linear metric learning is a widely used methodology to learn a dissimilarity function from a set of similar/dissimilar example pairs. Using a single metric may be a too restrictive assumption when handling heterogeneous datasets. Recently, local metric learning methods have been introduced to overcome this limitation. However, they are subjects to constraints preventing their usage in many applications. For example, they require knowledge of the class label of the training points. In this paper, we present a novel local metric learning method, which overcomes some limitations of previous approaches. The method first computes a Gaussian Mixture Model from a low dimensional embedding of training data. Then it estimates a set of local metrics by solving a convex optimization problem; finally, a dissimilarity function is obtained by aggregating the local metrics. Our experiments show that the proposed method achieves state-of-the-art results on four datasets.

Keywords: Dissimilarity learning, local metric learning, face recognition, nearest neighbor classification.

1 Introduction

An effective dissimilarity function is of great importance for many pattern recognition applications such as face verification, speech recognition or text categorization. As it might be hard to design such a function by hand for each task, especially when the input dimensionality is large, learning dissimilarity functions from labeled data has received a growing interest over the last years, see, for example, [4,7,24] and references therein.

A well studied class of dissimilarity functions is a linear metric of the form $(x_i - x_j)^\top M(x_i - x_j)$ where x_i and x_j are two data points which we wish to compare and M is a positive semidefinite matrix (PSD). This approach has been shown to be quite effective on various tasks but it suffers a strong limitation: it makes use of a single linear metric to compare data over all the input space which is inappropriate to handle heterogeneous data. This observation is the root of the development of local metric learning methods which adapt the dissimilarity function to the local specificities of the data. For illustrative purpose let us

D. Fleet et al. (Eds.): ECCV 2014, Part II, LNCS 8690, pp. 679–694, 2014.

consider two examples. It is well known that in digit classification some digits are easily mistaken for another such as "1" and "7" or "3" and "8", it seems therefore reasonable to reduce the number of misclassification to focus on different features to discriminate digits in the "1-7" region and in the "3-8" one. Our second example is face verification: should we put the emphasis on the very same features to compare two pictures of Caucasian males and two pictures of Asian females? Our answer is "no" and our experiments show that local metric learning improves the performance for these two applications.

Dissimilarity function can be employed in nearest neighbor classifiers or to take decisions based on the thresholding of the dissimilarity. Those two situations are different as the former depends only on the ranking of the nearest neighbors whereas the latter is concerned about the absolute meaning of the dissimilarity. In this work we propose a flexible local metric learning method called Large Margin Local Metric Learning (LMLML) which can be employed in both settings and can handle an arbitrary large number of classes. Its training procedure does not need one to know the class labels but only a set of pairs labeled "similar" (both points belong to the same class) or "dissimilar" (the two points belong to different classes). Our method computes a set of local metrics which are combined into an adaptive dissimilarity function with the help of a soft partitioning of the training data. The optimization of the local metrics is formulated as a convex problem which favors a large margin solution. The problem also involves a novel regularization term encouraging matrices which are close to a simple baseline solution. Our experiments show that LMLML outperforms or matches state-of-the-art results on various datasets.

After describing the related work in the next section we present LMLML and explain how to train the model in Section 3 and demonstrate its effectiveness on various datasets in Section 4. Finally, in Section 5 we summarize our findings.

2 Related Work

Global metric learning methods such as ITML [4], LDML [7] or LMNN [24] are precursors of this work. They formulate metric learning as the optimization of an objective function which decreases when the distances of similar pairs is made small while increasing those of dissimilar pairs. Some of these methods also include a regularization term which aims at limiting the risk of over-fitting.

Local metric learning has recently been investigated from several angles. It has sometimes been linked to semi-supervised clustering such as in [1] where labeled data are used to find local transformations of the data points in order to improve a clustering process. This kind of method cannot compute dissimilarity measures between pairs of never seen points which is the goal of our work.

Metric learning is often used in nearest neighbor schemes (NN) to perform multi-class classification. Several local metric learning algorithms have been developed to improve NN classifiers. Weinberger and Saul proposed an extension of LMNN to local metric (MM-LMNN [23]), in which a specific metric is associated to each class and all the metrics are jointly learned to optimize a classification criterion. LMNN has also been extended to a multi-tasks setting [5] where multiple

metrics are jointly learned [17]. Our work can be considered to be a generalization of these methods as it uses a weighted combination metrics instead of activating a single metric for each comparison (see Section 3.2).

KISSME [11] has also been extended to local metric in [12] where one KISSME metric is learned separately for each class. These class-specific metrics are averaged with a global one to limit the risk of over-fitting due to the fact that each metric might be learned using only a limited number of training samples. GLML [15] proposes to use local metric to limit the performance bias due to finite sampling using the class conditional probability distribution.

All the previously presented methods suffer from the same drawback, namely they need enough training samples per class to estimate the metrics and therefore cannot easily be employed when the number of classes is very large. To overcome this problem [22] introduces a local metric learning method based on finite number of linear metrics named PLML. The number of metrics is different from the number of classes and hence the method can scale to a larger number of classes. However this method is specifically designed for NN classification as it can only compute the dissimilarity of pairs for which at least one point is in the training set. This strongly limits the range of tasks PLML can deal with and for example prevent its use for face verification.

As we shall see in the next sections, our approach overcomes the limitations mentioned above and, as we demonstrate by a set of experiments, obtains better results than previous local metric learning methods.

3 Large Margin Local Metric Learning

This section describes the proposed method. We start by presenting our data preprocessing and then detail LMLML's model and each step of its optimization.

3.1 Data Preprocessing

A preprocessing is applied to the data before training the model. This step serves two purposes: first it reduces the dimensionality to speed up the computation for both training and testing, and second it reduces the noise thereby improving the overall performance of the algorithm.

Like most of the metric learning methods we first center the dataset and reduce the dimensionality to a n-dimensional space by PCA; n is often chosen so that 95% of the energy is preserved. Let $\mathcal{D} = \{(i, j)\}$ denote the index set of training pairs and let y_{ij} be a label which is equal to 1 if (x_i, x_j) is a similar pair and -1 otherwise. We compute the within-class scatter matrix defined by

$$S = U \left(\sum_{(i,j)\in\mathcal{D}|y_{ij}=1} (x_i - x_j)(x_i - x_j)^\top \right) U^\top \tag{1}$$

where U is the matrix formed by the n leading eigenvectors of the covariance matrix of the data, and then multiply the data by $S^{-1/2}$ to make the classes, on

average, isotropic. This transformation is known under different names such as *mapping in the intra-personal subspace* in face recognition [3] or *Within-Class Covariance Normalization (WCCN)* in the speaker recognition community [8]. The transformed data points are now $x' = S^{-1/2}U(x - m)$ where m is the mean of the data. Like in [3], we finally rescale each feature vector so that it has a unit L2-norm. To simplify the notation, whenever we mention a feature vector x in the reminder of the document, it in fact refers to the n-dimensional preprocessed vector $x'/\|x'\|_2$.

3.2 Model

The square distance associated with a pair of data points (x_i, x_j) using the matrix $M \in \mathcal{S}_+^n$ (the set of $n \times n$ PSD matrices) is defined by

$$d^2(x_i, x_j, M) = (x_i - x_j)^\top M(x_i - x_j). \tag{2}$$

In LMLML the metric is adapted to each pair so the matrix M is replaced by a matrix-valued function $\mathcal{M} : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathcal{S}_+^n$. It is defined, for every $x_i, x_j \in \mathbb{R}^n$, as a convex combination a $K + 1$ matrices

$$\mathcal{M}(x_i, x_j) = \sum_{k=0}^{K} w_k(x_i, x_j) M_k \tag{3}$$

where $w_k(x_i, x_j)$ are nonnegative weights which we define below. The smoothness of the function $\mathcal{M}$ is a very desirable property because it guarantees the dissimilarity function to be local and also because abrupt changes would make the thresholding of the dissimilarity more difficult to handle. In order to ensure this property, we use weights w_k which vary smoothly across the input space. As we want the dissimilarity function to be *local* it makes sense to use a soft partitioning of the input space to compute the weights $w_k(x_i, x_j)$ and we propose to rely on a Gaussian Mixture Model (GMM) with K components as follow:

$$w_k(x_i, x_j) = \begin{cases} \beta & \text{if } k = 0 \\ P(k|Vx_i) + P(k|Vx_j) & \text{otherwise} \end{cases} \tag{4}$$

where $V \in \mathbb{R}^{r \times n}$ is a dimensionality reduction matrix to be described in Section 3.3, β is a positive constant and $P(k|Vx)$ is the posterior probability that the point Vx has been generated by the Gaussian k. Notice that $\forall x_i, x_j \in \mathbb{R}^n, \sum_{k=0}^{K} w_k(x_i, x_j) = 2 + \beta$. The GMM has to be computed on a low dimensional space, typically less than 50, to ensure the smoothness of w_k because $P(k|Vx)$ tends not to be smooth otherwise. We illustrate this in Section 4.3. The GMM is trained using the standard Maximum Likelihood EM algorithm. Even if this procedure is totally unsupervised, the soft partitioning depends on the labels as V is learned in a supervised manner.

Thanks to the weights $w_k(x_i, x_j)$, each local metric has a strong influence only within a specific region, M_k has a large weight in $\mathcal{M}(x_i, x_j)$ if x_i or x_j is

strongly associated with Gaussian k and even more so if both are. In the face verification example mentioned in the introduction, the soft partitioning tends to roughly regroup faces by gender and ethnicity (see Section 4.4). Thus $\mathcal{M}(x_i, x_j)$ emphasizes those features which are most discriminative to compare people with similar gender/ethnicity than x_i and/or x_j.

The metric M_0 is associated with a constant weight and is therefore a global metric, it handles the part of the dissimilarity function which is common to the whole input space. The metrics M_k with $1 \leq k \leq K$ deal with the local adaptations of the dissimilarity function over the regions defined by the Gaussians of the GMM. Our model is a generalization of global metric and purely local metric as the parameter β allows to balance the influence of the global metric M_0 and the local metrics M_k in the matrix $\mathcal{M}(x_i, x_j)$. The larger K is, the more the model will be able to handle subtle local adaptations but the more it might overfit. Among values obtaining comparable performance during cross-validation, the smallest should be preferred because speed and memory occupancy for training and testing grow linearly with K. The impact of K on the performance is studied in Section 4.3. If $K = 0$ or $\beta \to \infty$ our model is equivalent to a global linear metric.

One could think that it is not necessary to add the M_0 metric in our model because the very same dissimilarity function can be written without M_0 by integrating it into the others M_k. However the actual formulation allows one to use all the training points to learn the part of the dissimilarity function which is common to the whole space and this leads to better generalization performance.

We summarize here the 4 steps of LMLML's training:

1. Data preprocessing (Section 3.1),
2. Optimization of V which maps the data into a low dimensional embedding (Section 3.3),
3. Training of the GMM and computation of the weights $w_k(x_i, x_j)$ using (4),
4. Optimization of the PSD matrices $M_0, \ldots, M_k$ (Section 3.4).

3.3 Low Dimensional Embedding

The weights $w_k(x_i, x_j)$ are based on the soft partitioning of the space derived from the GMM. Training the GMM on a discriminative low dimensional embedding is one of the key points of our method for two reasons. First, the low dimensionality favors the smoothness of weight functions. Second, for local metric to be effective it is needed that points which are hard to discriminate using a global metric have similar weights so that the local metrics focus on the locally discriminative information. To fulfill these two requirements the computation of the embedding is inspired by low rank global metric learning methods [20]. Let $V \in \mathbb{R}^{r \times n}$ denote the transformation matrix embedding the data into the low dimensional feature space of size r. We define the following loss function based on the hinge loss:

$$\ell_\gamma(y, z) = \max\left(0, 1 - \frac{y}{\gamma}(1 - z)\right) \tag{5}$$

where $y \in \{-1, 1\}$ is a label, z a dissimilarity value and γ the margin width parameter. We propose to find the transformation V by minimizing the objective function

$$J(V, \alpha) = \frac{1}{|\mathcal{D}|} \sum_{(i,j)\in\mathcal{D}} \ell_\gamma \left(y_{ij}, d^2(x_i, x_j, V^\top V)\right) + \lambda \|V^\top V - \alpha I\|_F \tag{6}$$

with respect to V and α, d^2 is the distance function defined by (2), $|\cdot|$ is the cardinality operator and λ is a parameter tuning the strength of the regularization. Thanks to the preprocessing step described in Section 3.1 the Euclidean distance is already a reasonable metric so, to limit over-fitting, the regularizer favors solutions close to the identity up to a scale factor parameter α. The scale factor being unknown, the optimization is also performed with respect to α.

The parameter γ is typically between 0.5 and 1: only the more difficult pairs impact the objective function when γ is small but a larger proportion of them does if γ is large. Its optimal value depends on how helpful easy pairs are to improve the performance on the part of the ROC curve we care about. It also depends on the size of the training set: larger values of γ are better with small training sets because when few pairs are available it is better not to discard too many of them even if they are not the most helpful ones. Both λ and γ have to be selected by cross-validation but can be set by looking only at the performance of low dimensional transformation alone. Objective functions similar to (6) have already been proposed in the literature [20,13], they mainly differ by the choice of the regularizer.

This optimization problem is not convex so there is no guarantee to find the global minimum but in practice we observed that the initialization of the optimization does not impact much the final performance. We use a stochastic mini-batch gradient descent to minimize $J(V, \alpha)$. At each iteration, it uses the gradient of $J_{\mathcal{D}'}(V, \alpha)$ which is the approximation of the objective function using only a randomly selected subset of size t of the training set $\mathcal{D}' \subset \mathcal{D}$. Its computation is straightforward and gives

$$\frac{\partial J_{\mathcal{D}'}(V, \alpha)}{\partial \alpha} = \frac{\alpha t - \|V\|_F^2}{\|V^\top V - \alpha I\|_F}, \tag{7}$$

$$\frac{\partial J_{\mathcal{D}'}(V, \alpha)}{\partial V} = \frac{1}{|\mathcal{D}'|} \sum_{(i,j)\in\mathcal{D}'} \frac{\partial \ell_\gamma \left(y_{ij}, d^2(x_i, x_j, V^\top V)\right)}{\partial V} + \lambda \frac{\partial \|V^\top V - \alpha I\|_F}{\partial V} \tag{8}$$

with

$$\frac{\partial \ell_\gamma \left(y_{ij}, d^2(x_i, x_j, V^\top V)\right)}{\partial V} = \begin{cases} 0 \text{ if } y_{ij}(1 - d^2(x_i, x_j, V^\top V)) < \gamma \\ \\ \frac{2y_{ij}V(x_i - x_j)(x_i - x_j)^\top}{\gamma} \text{ otherwise} \end{cases}, \tag{9}$$

$$\frac{\partial \|V^\top V - \alpha I\|_F}{\partial V} = \frac{2V(V^\top V - \alpha I)}{\|V^\top V - \alpha I\|_F}. \tag{10}$$

3.4 Local Metrics Optimization

LMLML training is a two steps procedure: we first learn the low dimensional transformation and the GMM allowing to compute the weights $w_k(x_i, x_j)$ as previously described, and then we optimize over the PSD matrices M_k considering the weights fixed. In this section, we explain how to achieve the latter step.

We define an objective function similar to (6) but adapted to local metrics:

$$H(\mathbf{M}, \boldsymbol{\alpha}) = \frac{1}{|\mathcal{D}|} \sum_{(i,j)\in\mathcal{D}} \ell_\gamma \left(y_{ij}, d^2(x_i, x_j, \mathcal{M}(x_i, x_j))\right) + \lambda' \sum_{k=0}^{K} \|M_k - \alpha_k I\|_F \quad (11)$$

where $\mathbf{M} = \{M_0, \ldots, M_K\}$, $\boldsymbol{\alpha} = \{\alpha_0, \ldots, \alpha_K\}$ is a vector of scale factors and λ' tunes the intensity of the regularization; λ' is usually very close to λ of (6) but can be tuned by cross-validation to optimize the performance of LMLML. We seek to minimize $H(\mathbf{M}, \boldsymbol{\alpha})$ with respect to $\mathbf{M}$ and $\boldsymbol{\alpha}$ under the constraint that $\forall k \in \{0, \ldots, K\}$, $M_k \in \mathcal{S}_+^n$.

When replacing $\mathcal{M}(x_i, x_j)$ by its definition (3) in the expression of the squared distance (2), we notice that $d^2(x_i, x_j, \mathcal{M}(x_i, x_j))$ is linear with respect to each of the matrices M_k so ℓ_γ is convex. Each term of the regularizer can be written as $\| \left[\mathrm{vec}(M_k)^\top, \alpha\right] A\|_2$ with the appropriate matrix $A \in \mathbb{R}^{(n^2+1)\times n^2}$. The composition of a convex function with a linear mapping is convex (see [2], 3.2.2) therefore $H(\mathbf{M}, \boldsymbol{\alpha})$ is jointly convex.

The optimization of a constrained problem can be slow but we can transform it into an unconstrained problem. We make the change of variable $M_k = L_k^\top L_k$ where $L_k \in \mathbb{R}^{n\times n}$, define the objective function H' such that $H'(\mathbf{L}, \boldsymbol{\alpha}) = H(\mathbf{M}, \boldsymbol{\alpha})$ where $\mathbf{L} = \{L_0, \ldots, L_K\}$ and optimize H'. This new problem is no longer convex but as it has been shown in [13], if we consider two functions f and h such that $f(L) = h(L^\top L)$ then every minimum of f corresponds to a minimum of h. The proof can easily be extended to the multi metric case and therefore, as H is convex, we can optimize the unconstrained problem H' without risking to be stuck in a non optimal local minimum. Notice that this reasoning does not apply to the minimization of (6) because the matrix V is rectangular whereas the L_k are square. Using a rectangular matrix is equivalent to adding a rank constraint on the metric and therefore makes the initial optimization problem non convex.

The optimization is also performed using a stochastic mini-batch gradient descent. Let $H'_{\mathcal{D}'}(\mathbf{L}, \boldsymbol{\alpha})$ be the approximation of the objective function using only a subset of size t of the training set $\mathcal{D}' \subset \mathcal{D}$, its gradient is

$$\frac{\partial H'_{\mathcal{D}'}(\mathbf{L}, \boldsymbol{\alpha})}{\partial \alpha_k} = \frac{\alpha_k t - \|L_k\|_F^2}{\|L_k^\top L_k - \alpha_k I\|_F}, \quad (12)$$

$$\frac{\partial H'_{\mathcal{D}'}(\mathbf{L}, \boldsymbol{\alpha})}{\partial L_k} = \frac{1}{|\mathcal{D}'|} \sum_{(i,j)\in\mathcal{D}'} \frac{\ell_\gamma \left(y_{ij}, d^2(x_i, x_j, \mathcal{M}(x_i, x_j))\right)}{\partial L_k} + \lambda' \frac{\partial \|L_k^\top L_k - \alpha_k I\|_F}{\partial L_k} \quad (13)$$

with

$$\frac{\ell_\gamma\left(y_{ij}, d^2(x_i, x_j, \mathcal{M}(x_i, x_j))\right)}{\partial L_k} = \begin{cases} 0 \text{ if } y_{ij}(1 - d^2(x_i, x_j, \mathcal{M}(x_i, x_j))) < \gamma \\ \frac{2y_{ij}w_k(x_i,x_j)L_k(x_i-x_j)(x_i-x_j)^\top}{\gamma} \text{ otherwise} \end{cases} \tag{14}$$

$$\frac{\partial \|L_k^\top L_k - \alpha_k I\|_F}{\partial L_k} = \frac{2L_k(L_k^\top L_k - \alpha_k I)}{\|L_k^\top L_k - \alpha_k I\|_F}. \tag{15}$$

Note that in LMLML, the L_k are jointly optimized: all the metrics impact the value of $\mathcal{M}(x_i, x_j)$ and therefore the gradient of each L_k.

3.5 Parameters Setting

The regularization strength λ' and the number of local metrics K are the two key parameters of LMLML. They have to be set by cross-validation for each dataset, the others parameters don't need to be fine tuned. The dimensionality after PCA n is common to most metric learning algorithm and is usually chosen so that 95% of the energy is preserved. Parameter γ is not very sensitive, it takes value in $[0.5, 1]$ and should be close to 0.5 on big datasets and closer to 1 on smaller ones. Among the additional parameters added specifically to deal with the local metrics, namely K, r and β, only K needs to be tuned, we have set $r = 30$ and $\beta = 1.5$ in all our experiments.

4 Experiments

The set of experiments presented in this section demonstrates the performance of LMLML on various datasets and provides some insights into the method.

4.1 Datasets and Setup

MNIST. Handwritten digits classification has been widely used to assess the performance of dissimilarity functions for classification. The MNIST[1] dataset is composed of 70000 images of size 28×28, 60000 for training and 10000 for testing. We used the same setup as in [22] to compare the performance with other dissimilarity measures. The PCA is computed directly on the pixel values and 164 dimensions are kept after the PCA to retain 95% of the energy. The classification is performed using a simple nearest neighbor classifier.

FRGC. Face verification is a very popular application of dissimilarity function learning. The task consists in determining whether two images depict the same person, usually by thresholding a dissimilarity measure. The identities used for testing are not included in the training set. FRGC Experiment 1 [18] is a face

[1] `http://yann.lecun.com/exdb/mnist/`

dataset of more than 15000 images of more than 500 people. The pose of the subject and the illumination have been controlled during the acquisition so compared to datasets like Labeled Faces in the Wild presented hereafter this dataset is fairly easy. However, on this type of dataset, the interest is focused on the verification rate at low false positive rates (1% and below). This is a realistic setting for many security applications of face recognition (like smartphone unlocking or passport check at the border) where a false accept is a security breach and therefore must be very rare. After aligning the images using the eyes locations, we computed UoCTTI HOG descriptors [6] extracted using the *VLFeat* library [21] to obtain 6076-dimensional feature vectors. We retained 700 dimensions after the PCA and trained all the dissimilarity functions we compare on these vectors.

LFW. Label Faces in the Wild (LFW) [9] is another face verification dataset. It is composed of 13233 images of 5749 people taken from *Yahoo! News* in wide range of acquisition conditions (pose, illumination, expression, age, *etc.*) therefore considered to be challenging. We have followed the same feature extraction procedure as [3]: the "aligned" image are first cropped to 150×80 to remove most of the background, then descriptors composed of histograms of Local Binary Patterns [16] are extracted and their dimensionality is reduced to 300 by PCA. The within-class whitening is also activated on LFW. As discussed in Section 4.2 the choice of the training pairs has a great impact on the overall performance of dissimilarity function methods. To get rid of this bias we report the performance on LFW in the Image Restricted setting where only a limited number of labeled training pairs are provided but not the identity information associated with each image making impossible the use of all possible pairs. We followed the recommended 10-folds cross-validated experiment for evaluation [9] and selected the parameters and the threshold using a 9-folds cross-validation scheme for each training.

Reuters. Finally we demonstrate the performance of LMLML on the text categorization dataset Reuters-21578 R52[2]. It consists of 9100 text documents which appeared on the Reuters newswire in 1987, 6532 in the training test and 2568 in the testing set. Each text belongs to one of the 52 topics and every topic has at least one text in the training set and one in the testing set. The classes are very unbalanced as some topics have more than 1000 text documents whereas others have just a few. Each text is described as a histogram of word occurrence spanning 5180 terms. A very large number of dimensions should be kept after the PCA to preserve 95% of the energy but to speed up the experiments we kept only the first 100 dimensions retaining only 62% of the energy.

[2] `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

4.2 Choosing Training Pairs from Class Labels

Many dissimilarity function learning methods consist in optimizing an objective function depending on pairs of feature vectors, see for example [4,7,22,3,13]. The number of possible pairs grows quadratically with the number of training points therefore it is often impracticable to use all of them because the training would require a prohibiting amount of time. The choice of the training pairs has a huge impact on the performance of the method. On some datasets like LFW in the Image Restricted setting the pairs are provided with the dataset but most of the time the pairs have to be created from the class labels, in this section we give some guidelines about how to choose the training pairs.

When the dissimilarity function is employed for nearest neighbor classification like on the MNIST and Reuters datasets, the training pairs should consist of neighboring points because nearest neighbor classifiers base their decision only from such points. We propose to proceed as follow: for each data point x, create q similar pairs, formed by x and each of its q closest neighbors of the same class, and q dissimilar pairs, formed by x and each of its q closest neighbors of a different class. This results in $2q$ training pairs per data point. For both MNIST and Reuters we used $q = 5$.

Dissimilarity measures are also often thresholded to take a decision such as in face verification application. The choice of the threshold leads to a specific trade-off between false positive error and verification rates and this choice depends on the target application. On a dataset like FRGC the number of possible similar pairs is limited and all of them can be used during training but a selection must be made for the dissimilar ones. As people are mainly interested in performance at low false positive rates on such datasets, the training set should include a large number of dissimilar pairs. To speed up the training, we propose a simple hard dissimilar pairs mining scheme. We first randomly pick a number of dissimilar pairs equal to the number of similar pairs and train our model with this set. Then we compute the dissimilarity for a large number of dissimilar pairs and select the 5 or 10% hardest pairs to learn the metrics again. This step could be repeated many times but in practice we observed little improvement after the first iteration.

4.3 Results on MNIST

We argued in Section 3.2 that the dimensionality of the data needs to be reduced before computing the GMM in order to ensure the smoothness of the posterior probability $P(k|x)$ with respect to x and therefore of the weights w_k and the function $\mathcal{M}$. We performed an experiment on MNIST to support our claim. We computed low dimensional embeddings of several dimensionalities using the procedure described in Section 3.3 and for each performed the following operations:

1. Estimation of the GMM parameters (for this experiment we have arbitrary chosen $K = 5$),

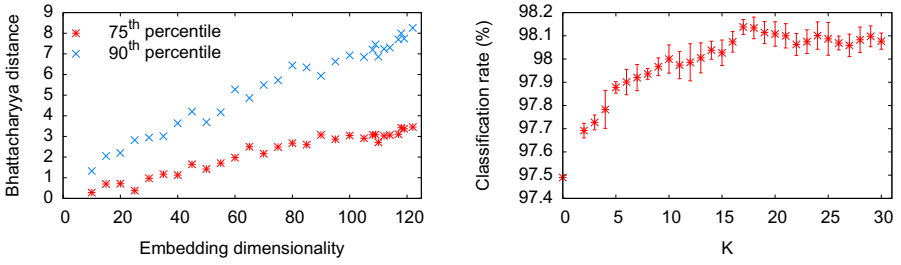
2. Computation of the posterior probability distribution $P(\cdot|x)$ for every x in the training set,
3. Computation of the Bhattacharyya distance

$$d_B\left(P(\cdot|x_i), P(\cdot|x_j)\right) = -\log \sum_k \sqrt{P(k|x_i)P(k|x_j)} \tag{16}$$

between the distribution associated with each x and those of its 3 nearest neighbors.

Figure 1a shows the 75^{th} and 90^{th} percentiles of the computed Bhattacharyya distance function of the dimensionality of the embedding. We can see that larger embedding dimensionality leads to larger Bhattacharyya distance between nearby points which means less smooth weights w_k and therefore justify our choice of a low dimensional embedding. When $r = 30$, the largest weight accounts for less than 80% of the total for 18% of MNIST samples.

We also studied on this dataset the impact of the parameter K, the number of local metrics used in LMLML. As the training of the GMM only finds a local minimum we performed the training 10 times for each value of K and report the mean and standard deviation of the classification rate on Figure 1b. We see that the local metrics significantly outperform the global one ($K = 0$) even if only 2 metrics are used. The classification rate increases for K up to 17 and stays stable afterwards. The speed and memory occupancy for training and testing grows linearly with K so smaller values should be favored. We also notice that LMLML's overall performance has little sensitivity to the local minima proneness of the GMM as the average standard deviation is only 0.015%.



(a) (b)

Fig. 1. (a) Percentiles of Bhattacharyya distance between neighbors function of the embedding dimensionality. (b) Impact of K on LMLML's performance.

On the MNIST dataset we compare LMLML with 6 others methods designed for nearest neighbor classification using the same features: 2 global metric learning algorithms LMNN [24] and BoostMetric [19], 3 local metric learning methods MM-LMNN [23], GLML [15] and PLML [22] and a multi-class SVM with one-against-all strategy (the best kernel has been chosen by inner cross-validation). The performance of the other methods have been taken from [22]. Table 1 reports the results. It is worth noticing that the idea of local metric learning needs

to be carefully handled to be effective, MM-LMNN and GLML obtain worse results than the global metric learning methods. A possible explanation is that they learn one metric per class and fail to share the information among the classes. LMLML ($K = 19$ set by cross-validation) has an accuracy of 98.10% and outperforms all the other methods including non linear SVM.

Table 1. Classification Rates on MNIST

Local Metric Learning & SVM		Global Metric Learning	
Method	Class. Rate	Method	Class. Rate
LMLML $K = 19$	98.10%	LMLML $K = 0$	97.49%
SVM [22]	97.62%	LMNN [24]	97.30%
PLML [22]	97.30%	BoostMetric [19]	96.03%
MM-LMNN [23]	93.24%		
GLML [15]	84.00%		

4.4 Results on FRGC

To perform face verification a method needs to be able to compute a dissimilarity with pairs of never seen points so we cannot compare LMLML to the local metric learning methods presented in Section 2. Figure 2 shows the ROC curves of LMLML, KISSME [11], ITML [4] and LDML [7]. We used the code available on their respective author's website and cross-validated their parameters in the same way we did with LMLML following the authors' recommendations. Our approach obtains better verification rates at all false positive rates. We also experimented LMNN but it works poorly on this dataset. This result is not surprising as LMNN is designed to find a good metric for nearest neighbor classification but not for thresholding.

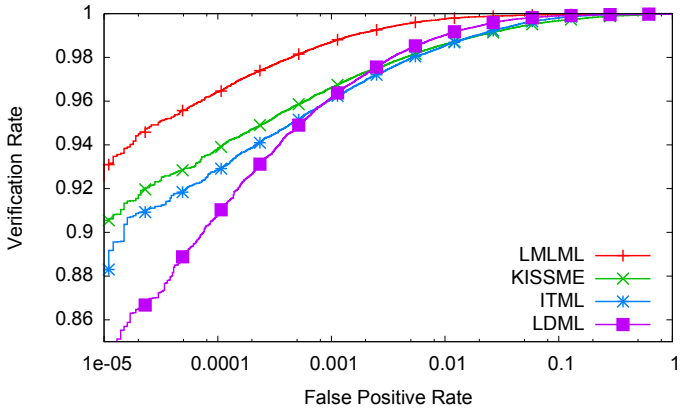


Fig. 2. ROC curves on FRGC

In order to gain more insights into the good performance achieved by our method we observed the distribution of the faces among the different Gaussians of the GMM. Each line of Figure 3 shows faces for which the posterior probability is greater than 0.7 for a specific Gaussian. Thanks to the use of our discriminative low dimensional embedding, the GMM capture interesting properties of the faces: the first group is mostly populated of Asian people (both males and females), the second of Caucasian females, the third and fourth of Caucasian males and the last one of Asian females. This grouping is totally unsupervised as no gender or ethnicity information has been given to the algorithm and allows LMLML to adapt the dissimilarity function to the specificities of these groups and contribute to its good performance on FRGC.

Fig. 3. Samples of faces belonging to the different Gaussians of the GMM, each line is constituted of faces for which $P(k|Vx) > 0.7$ for a specific Gaussian

4.5 Results on LFW

The best methods on LFW rely on fine tuning of the feature extraction: combination of several image descriptors, detection of numerous feature points, use of multiple alignments of the images. To fairly evaluate our approach we followed the feature extraction procedure described in [3,14] and compare our results to those given in these papers in Table 2. CSML's accuracy is copied from [14] and all the others come from [3].

LMLML obtains the 2$^{\text{nd}}$ best performance with 0.8613 just behind Sub-SML which got 0.8673. However, the parameter selection process consistently chooses

Table 2. Accuracy on LFW

Method	Accuracy	Method	Accuracy
LMLML	0.8613 ± 0.0053	ITML [4]	0.7998 ± 0.0039
Sub-SML [3]	0.8673 ± 0.0053	Sub-ITML [3]	0.8398 ± 0.0048
CSML [14]	0.8557 ± 0.0052	LDML [7]	0.8065 ± 0.0047
KISSME [11]	0.8337 ± 0.0054	Sub-LDML [3]	0.8227 ± 0.0058
DML-eig [25]	0.8228 ± 0.0041	SILD [10]	0.8007 ± 0.0135

$K = 0$ for all the folds meaning that local metrics does not help on this dataset. The main explanation for this observation might be that the major challenge of LFW is the wide intra-class variability and this issue is not addressed by local metric learning as local metrics mainly help to discriminate similar pairs from dissimilar ones which are composed of close by feature vectors.

4.6 Results on Reuters

As we noted before this dataset is composed of a large number of classes and is very unbalanced. This makes local metric learning methods which learn one metric per class such as MM-LMNN [23] or GLML [15] not well suited. We compared LMLML with one global metric learning method: LMNN, and one local learning metric method: PLML. We used the code downloaded from their respective author's website and cross-validated their parameters. The results are the following: LMLML $K = 3$: 89.03%, LMLML $K = 0$: 88.75%, LMNN: 88.87% and PLML 87.39%. LMLML obtains the best performance but is only slightly better than LMNN. Once again these results show that handling local metric can be tricky as the global metric learning LMNN outperforms PLML.

5 Conclusion

In this paper, we have introduced a new local metric learning algorithm. The data are embedded into a discriminative low dimensional space to compute a soft partitioning which allows to define a smooth locally adapting dissimilarity function. Our method overcomes the limitations of previous local metric learning methods, it is as flexible as global metric learning methods and can therefore be applied to wide variety of scenarios. The good performance of LMLML has been demonstrated on four different datasets including the popular and challenging LFW. In the future, LMLML could be extended to a more generic multi metrics method by performing the soft partitioning with respect to other criteria such as image quality. Furthermore, an interesting direction for future research is to integrate the soft partitioning and the learning of the local metrics into a single optimization problem.

References

1. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML, pp. 11–18 (2004)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
3. Cao, Q., Ying, Y., Li, P.: Similarity metric learning for face recognition. In: ICCV (2013)
4. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML, pp. 209–216 (2007)
5. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: SIGKDD, pp. 109–117 (2004)
6. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI 32(9), 1627–1645 (2010)
7. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? metric learning approaches for face identification. In: ICCV, pp. 498–505 (2009)
8. Hatch, A.O., Kajarekar, S., Stolcke, A.: Within-class covariance normalization for svm-based speaker recognition. In: ICSLP, pp. 1471–1474 (2006)
9. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst (2007)
10. Kan, M., Shan, S., Xu, D., Chen, X.: Side-information based linear discriminant analysis for face recognition. In: BMVC, pp. 1–12 (2011)
11. Köstinger, M., Hirzer, M., Wohlhart, P., Roth, P.M., Bischof, H.: Large scale metric learning from equivalence constraints. In: CVPR, pp. 2288–2295 (2012)
12. Köstinger, M., Roth, P.M., Bischof, H.: Synergy-based learning of facial identity. In: Pinz, A., Pock, T., Bischof, H., Leberl, F. (eds.) DAGM and OAGM 2012. LNCS, vol. 7476, pp. 195–204. Springer, Heidelberg (2012)
13. Maurer, A.: Learning similarity with operator-valued large-margin classifiers. JMLR 9, 1049–1082 (2008)
14. Nguyen, H.V., Bai, L.: Cosine similarity metric learning for face verification. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part II. LNCS, vol. 6493, pp. 709–720. Springer, Heidelberg (2011)
15. Noh, Y.K., Zhang, B.T., Lee, D.D.: Generative local metric learning for nearest neighbor classification. In: NIPS, pp. 1822–1830 (2010)
16. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. PAMI 24(7), 971–987 (2002)
17. Parameswaran, S., Weinberger, K.Q.: Large margin multi-task metric learning. In: NIPS, pp. 1867–1875 (2010)
18. Phillips, P.J., Flynn, P.J., Scruggs, T., Bowyer, K.W., Chang, J., Hoffman, K., Marques, J., Min, J., Worek, W.: Overview of the face recognition grand challenge. In: CVPR, pp. 947–954 (2005)
19. Shen, C., Kim, J., Wang, L., van den Hengel, A.: Positive semidefinite metric learning with boosting. In: NIPS, pp. 1651–1659 (2009)
20. Simonyan, K., Parkhi, O.M., Vedaldi, A., Zisserman, A.: Fisher vector faces in the wild. In: BMVC (2013)

21. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008), http://www.vlfeat.org/
22. Wang, J., Kalousis, A., Woznica, A.: Parametric local metric learning for nearest neighbor classification. In: NIPS, pp. 1610–1618 (2012)
23. Weinberger, K., Saul, L.: Fast solvers and efficient implementations for distance metric learning. In: ICML, pp. 1160–1167 (2008)
24. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. JMLR 10, 207–244 (2009)
25. Ying, Y., Li, P.: Distance metric learning with eigenvalue optimization. JMLR 13, 1–26 (2012)