Regression-based Metric Learning

Panagiotis Moutafis, Mengjun Leng, and Ioannis A. Kakadiaris
Computer Science, University of Houston, Houston, Texas 77004
{pmoutafis, mleng2, ioannisk}@uh.edu

***Abstract*—Existing distance metric learning methods define an objective function and seek a distance metric (or equivalently a projection) that minimizes it. In this paper, we propose a different approach that illustrates how to formulate distance metric learning as a regression problem. First, the objective function is minimized to learn target representations. Then, a regression method is employed to learn a projection that maps the input to the target representations. This global projection function is the single output of the proposed algorithm. Our contribution is a different perspective on how to train a distance metric learning algorithm. The advantages are: (i) this approach has the potential to simplify the optimization process; and (ii) it allows researchers to leverage the power of existing regression methods and those to be invented. Experimental results on several publicly available datasets illustrate that the proposed framework can learn a distance metric with discriminative properties.**

I. Introduction

Many applications can benefit from a similarity measure that accurately reflects what is considered to be "similar" or "dissimilar". However, the notion of similarity varies depending on the application. For example, different metrics should be used for identity versus gender classification. Distance metric learning methods address this problem by learning a dissimilarity measure (or equivalently a projection function) tailored to the task of interest. Such methods offer many advantages over traditional classification approaches including: (i) they naturally generalize to unseen classes without retraining; (ii) they are suitable for multi-class problems with few samples per class; and (iii) they are suitable for multimodally distributed data. The intuition behind most distance metric learning methods is the same - distances between samples with the same label should be "small" while distances between samples with different labels should be "large". Existing approaches propose different objective functions to capture this semantic notion. The objective function usually has a single model parameter (i.e., the distance metric to be learned) subject to a set of constraints. Hence, the optimization is cast as a search for a feasible solution that minimizes the given objective function. Depending on the objective function and the constraints used, though, solving the optimization problem might not be straightforward. For example, non-linear problems are known to be intrinsically harder to solve than linear programming problems. Being able to solve such problems is important because a non-linear distance metric can often yield better accuracy compared to a linear one, even when they are both learned using the same objective function [14], [6].

In this paper, we illustrate how to cast distance metric learning as a regression problem. Unlike existing methods, the objective function is minimized with the goal of learning *target representations* and not a distance metric. Then, a regression method is employed to map the input to the target representations learned. The global projection function learned by the regression method employed is the single output of the proposed approach. In certain situations, this approach can simplify the optimization problem (e.g., learning a non-linear metric learning). We call our method Regression-based Metric Learning (RBML) and offer an overview in Fig. I. The performance of RBML is bounded by the capabilities of the regression method employed. To illustrate how this alternative way can be used to train a distance metric, we propose an objective function that relies on local relationships in the data inspired by the Large Margin Nearest Neighbor (LMNN) [14]. Deriving the target representation for one sample at a time is easy as the corresponding optimization problem is convex. To reduce the time complexity, an alternative geometric construction is proposed. Even though this solution is not guaranteed to find the global optimum for each sample, it appears to work well in practice. The target representations learned, denote the desired output should an ideal distance metric existed. As a proof of concept, a naive regression approach was employed to map the input to the target representations learned. An executable version of our code will be available upon request. Experimental results on publicly available datasets illustrate the effectiveness of the proposed geometric construction and provide evidence that RBML can be used to learn a distance metric with discriminative properties for a variety of tasks.

The rest of the paper is organized as follows: in Sec. 2 we review related work; in Sec. 3 the RBML is described and key aspects are discussed; in Sec. 4 the experimental evaluation is presented; and Sec. 5 concludes the paper.

II. Related Work

In this section, we offer an overview of the methods to be used in the experimental evaluation. Bi *et al.* [3] proposed the Cayley-Klein (CK) metric learning as an alternative way to the widely used Mahalanobis distances. The main idea is to seek a distance metric on a non-euclidean space using a cross-ratio. Weinberger *et al.* [14] proposed a Large Margin Nearest Neighbor (LMNN) approach that penalizes large distances between neighbors of the same class and small distances of samples of different class that violate a predefined margin. Kedem *et al.* [6] proposed a non-linear extension of LMNN. Specifically, the Gradient-Boosting LMNN (GB-LMNN) employs gradient-boosted trees to learn a non-linear mapping. For a comprehensive overview of distance metric learning methods

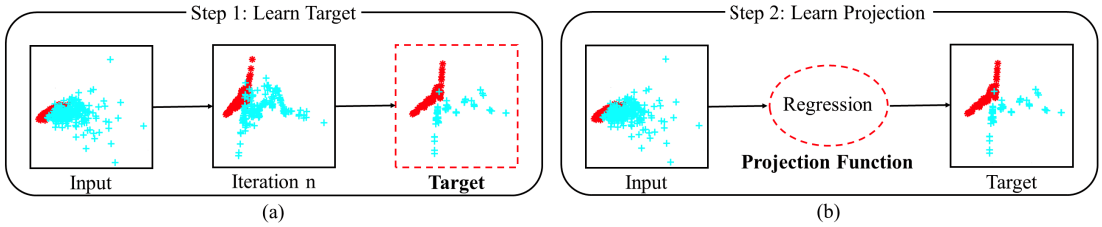

Fig. 1. Overview of the proposed approach which we call RBML. The two shapes (i.e., cross and star) denote two classes. The dashed lines and bold captions indicate what is learned in each step. (a) The objective function (i.e., Eq. (1)) is iteratively minimized to learn target representations (i.e., Target); (b) A regression method is employed to learn a projection function that maps the input to the target, where the input is used as the predictor variables and the target representations learned in the previous step are used as the response variables. The projection function learned in this step is the single output of RBML.

we refer interested readers to existing surveys [8], [10], [2]. The goal of this work is not to outperform all other methods. Instead, we seek to contribute an alternative way of learning distance metrics by simplifying the optimization process for certain cases. Therefore, we focus on two algorithms that minimize the same objective function to learn a linear and a non-linear distance metric (i.e., LMNN and GB-LMNN, respectively). We use the proposed method to minimize a similar objective function and we investigate how our non-linear distance metric compares to the existing approach.

III. Regression-based Metric Learning

In this section, we describe the proposed Regression-based Metric Learning (RBML) approach. Specifically, our objective is to learn a projection function that maps the input to a space where samples with the same class label are "closer" while samples with different class labels are "far away". The training process is divided into two steps: (i) minimize the objective function to learn target representations; and (ii) employ a regression model to learn a projection function $\mathcal{L}$ that maps the input to the target representations learned. Henceforth, lowercase bold letters denote vectors and uppercase bold letters denote matrices. In particular, $\mathbf{x}_i$ is an n^{th} dimensional data point where i denotes the sample index and $\mathbf{X}$ is a matrix of s such samples. The target representations learned are denoted by $\mathbf{x}_i^*$ and $\mathbf{X}^*$, respectively. The associated label for each sample is denoted by $y_i \in \mathbb{Z}$, where $\mathbb{Z}$ is the set of integers, and the vector with the labels for all the samples is denoted by $\mathbf{y}^\top$. We begin by describing the objective function.

A. Objective function

We use an objective function $\mathcal{C}$ that comprises two parts, $\mathcal{T}(\mathbf{X})$ and $\mathcal{H}(\mathbf{X})$. The trade-off between these terms is determined by the parameter $\alpha \epsilon [0, 1]$ as follows:

$$\mathcal{C}(\mathbf{X}) = (1-\alpha)\mathcal{T}(\mathbf{X}) + \alpha\mathcal{H}(\mathbf{X}) \ . \tag{1}$$

Target Neighbors: The term $\mathcal{T}(\mathbf{X})$ penalizes large distances between each sample i and its k-nearest neighbors with the same label (i.e., target neighbors). Specifically, for a given sample $\mathbf{x}_i$, the function $\delta_T(i,j)$ equals 1 if j is a target neighbor of i. Otherwise, $\delta_T(i,j) = 0$. The function $\delta_T(i,j)$ is not commutative. The cost incurred by the *target neighbors* is defined as $\mathcal{T}(\mathbf{X}) = \sum_{i=1}^{s} \mathcal{T}_v(\mathbf{x}_i)$, where

$$\mathcal{T}_v(\mathbf{x}_i) = \sum_{j=1}^{s} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \delta_T(i,j) \ . \tag{2}$$

Hinge Loss: The term $\mathcal{H}(\mathbf{X})$ imposes a penalty each time a sample with a different label violates a predefined margin (i.e., impostor). Specifically, for each sample $\mathbf{x}_i$ we define the margin m_i as the distance to the more distant target neighbor multiplied by a constant $\beta \geq 1$. That is,

$$m_i = \beta * \max_j \|\mathbf{x}_i - \mathbf{x}_j\|^2 \delta_T(i,j) \ . \tag{3}$$

For a given sample $\mathbf{x}_i$, the function $\delta_I(i,j)$ equals 1 if j is an impostor (i.e., $\|\mathbf{x}_i - \mathbf{x}_j\|^2 < m_i$ and i, j have different labels). Otherwise, $\delta_I(i,j) = 0$. The function $\delta_I(i,j)$ is not commutative. The cost incurred by the *impostors* is defined as $\mathcal{H}(\mathbf{X}) = \sum_{i=1}^{s} \mathcal{H}_v(\mathbf{x}_i)$, where

$$\mathcal{H}_v(\mathbf{x}_i) = \sum_{j=1}^{s} \left(m_i - \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \delta_I(i,j) \ . \tag{4}$$

B. Target representations

In this section, we describe two alternative ways to learn *target representations*.

The objective function $\mathcal{C}$ (Eq. 1) is not convex with respect to $\mathbf{X}$. However, it becomes convex if we solve $\mathcal{C}(\mathbf{X})$ for a given sample $\mathbf{x}_i$ and fix the rest. To denote this case, the term $\mathcal{C}_v(\mathbf{x}_i)$ is used. The function $\mathcal{C}_v(\mathbf{x}_i)$ is defined as the sum of convex functions (i.e., euclidean distance and max). As a result, $\mathcal{C}_v(\mathbf{x}_i)$ is also convex and existing methods can be used to minimize it. In our implementation, we used the Nelder-Mead simplex method [9]. Even though it is not guaranteed to converge to the global minimum, it has been found to work well in practice [7]. Also, it does not rely on numerical or analytic gradients which are expected to reduce its computational complexity. The version of our algorithm that employs the Nelder-Mead method is denoted by RBML-NM.

Despite being more efficient than other optimization methods, the Nelder-Mead still scales poorly with the number of samples and the dimensionality of the features. To address this problem, we propose a geometric construction that was empirically found to minimize $\mathcal{C}_v(\mathbf{x}_i)$ both efficiently and

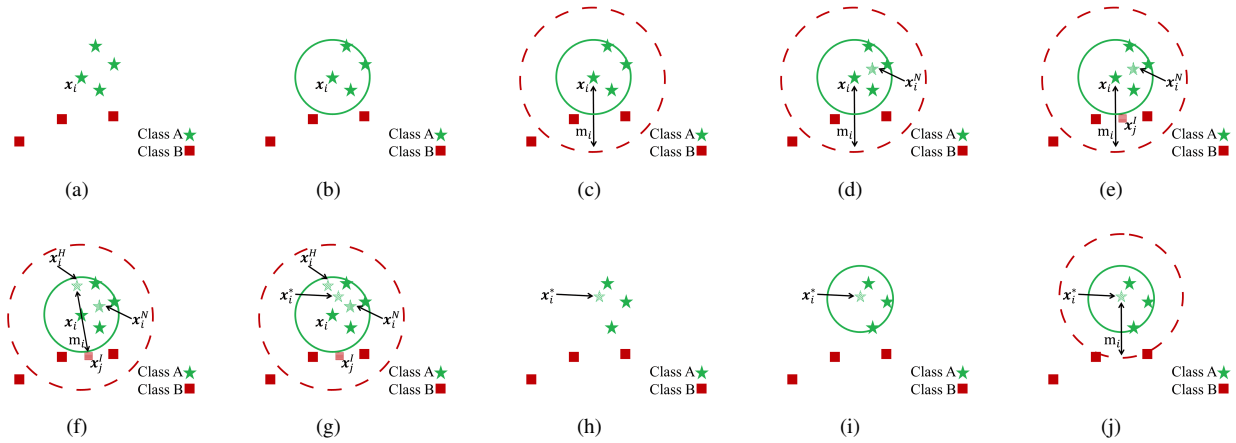

Fig. 2. Overview of the proposed geometric construction proposed to learn the target representations. (a) The two shapes (i.e., squares and stars) denote two classes and $\mathbf{x}_i$ denotes a given sample; (b) the circle denotes the $\max_j \|\mathbf{x}_i - \mathbf{x}_j\|^2 \delta_T(i,j)$ term in Eq. (3); (c) the outer circle (i.e., dashed line) denotes the margin m_i (i.e., Eq. (3)) and the square-shaped points inside it are two impostors; (d) the "new" star-shaped point denotes the target representation $\mathbf{x}_i^N$ that minimizes $\mathcal{T}_v$ (i.e., Eq. (2)); (e) the "new" square-shaped point denotes the "average impostor" $\mathbf{x}_j^I$ (i.e., Eq. (6)); (f) the "new" star point denotes $\mathbf{x}_i^H$ that minimizes $\mathcal{H}'_v$ (i.e., Eq. (7)); (g) the "new" star point denotes $\mathbf{x}_i^*$ as computed by Eq. (9); (h) the original point $\mathbf{x}_i$ and the other supporting data points are removed; (i) the new $\max_j \|\mathbf{x}_i - \mathbf{x}_j\|^2 \delta_T(i,j)$ term in Eq. (3) is computed using the target representation $\mathbf{x}_i^*$; (j) the outer circle (i.e., dashed line) denotes the new margin m_i (i.e., Eq. (3)) which does not yield any impostors.

effectively. In particular, $\mathcal{T}_v(\mathbf{x}_i)$ (Eq. (2)) and $\mathcal{H}_v(\mathbf{x}_i)$ (Eq. (4)) are treated separately. An overview of the proposed construction is offered by Fig. (2). The *target representation* $\mathbf{x}_i^N$ that minimizes the cost occurred by target neighbors is computed by minimizing Eq. (2):

$$\mathbf{x}_i^N = \sum_{j=1}^{s} \mathbf{x}_i \delta_T(i,j) \ . \tag{5}$$

To compensate for scaling problems and geometric consistency, the target representation $\mathbf{x}_i^N$ is divided by the number of target neighbors (i.e., k or $\sum_{j=1}^{s} \delta_T(i,j)$). In other words, $\mathbf{x}_i^N$ is computed as the "average of target neighbors".

To simplify our construction, we minimize $\mathcal{H}_v(\mathbf{x}_i)$ with respect to the "average impostor" denoted by $\mathbf{x}_j^I$. That is:

$$\mathbf{x}_j^I = \frac{\sum_{j=1}^{s} \mathbf{x}_j \delta_I(i,j)}{\sum_{j=1}^{s} \delta_I(i,j)} \ . \tag{6}$$

The updated hinge loss $\mathcal{H}'_v(\mathbf{x}_i)$ is defined as:

$$\mathcal{H}'_v(\mathbf{x}_i) = max\big(0, m_i - \|\mathbf{x}_i - \mathbf{x}_j^I\|^2\big) \ . \tag{7}$$

The function max is used because the hinge loss should be activated only when the distance $\|\mathbf{x}_i - \mathbf{x}_j^I\|^2$ is less than m_i. In the original formulation (i.e., Eq. (4)) this is achieved using the term $\delta_I(i,j)$. The target representation $\mathbf{x}_i^H$ that minimizes $\mathcal{H}'_v$ is computed as follows:

$$\mathbf{x}_i^H = \mathbf{x}_i + m_i \frac{(\mathbf{x}_i - \mathbf{x}_j^I)}{\|(\mathbf{x}_i - \mathbf{x}_j^I)\|^2} \ . \tag{8}$$

Hence, $\|\mathbf{x}_i^H - \mathbf{x}_j^I\|^2 = m_i$. Finally, $\mathbf{x}_i^*$ is computed as:

$$\mathbf{x}_i^* = (1-\alpha)\mathbf{x}_i^N + \alpha \mathbf{x}_j^H \ . \tag{9}$$

The version of our algorithm that relies on the proposed geometric construction is denoted by RBML-GE.

C. Algorithmic Implementation

In this section, the proposed algorithm is developed and the corresponding implementation details are discussed. An overview is provided by Algorithm (1).

The parameters that need to be configured are: (i) number of target neighbors k; (ii) trade-off weight between target neighbors and impostors α; and (iii) margin constant β. These values can be optimized by using cross-validation or defined arbitrarily by the user. The default values recommended are $k = 3$, $\alpha = 0.5$, and $\beta = 2$.

Line 1: To learn the target representations $\mathbf{x}_i^*$, Eq. (9) is successively employed. That is, after each iteration is completed, $\mathbf{x}_i^*$ is used to compute Eq. (9) until the convergence criterion has been met. In this paper, the classification accuracy was used to indicate whether the algorithm has converged. Specifically, the while loop is stopped when the k-nearest neighbor (k-NN) classification accuracy does not increase between any two successive updates or when it reaches 100%.

Line 3: To speed up the optimization process, the $\mathbf{x}_i^*$ updates can be performed using parallel processing for different samples. Algorithm (1) can perform dimensionality reduction if a reduced representation $\mathbf{X}' \epsilon \mathbb{R}^{n' \times s}$ is used in Eq. (9) instead of the original data $\mathbf{X} \epsilon \mathbb{R}^{n \times s}$ (i.e., $n' \leq n$).

Line 5: After the end of each iteration of the while loop, the target neighbors, impostors, and margins are redefined using the updated samples $\mathbf{x}_i^*$.

Line 7: To learn the mapping from $\mathbf{X}$ to $\mathbf{X}^*$, any regression method can be employed. In this paper, a naive *single-output non-linear regression* approach is adopted. Specifically, one regression model is learned for each dimension of the target domain independently using Random Regression Forests [4]. That is, n independent regression models are learned that map

Algorithm 1 RBML Training

Input: Data matrix $\mathbf{X}$ and labels vector $\mathbf{y}^\top$
Output: Projection function $\mathcal{L}$

Target representations
while convergence criterion has not been met **do**
  for $i = 1, \ldots, s$ **do**
    Update $\mathbf{x}_i^*$ according to Eq. (9).
  end for
  Update δ_T, m_i, and δ_I.
end while
Regression
Learn $\mathcal{L}$ by regressing $\mathbf{X}$ to $\mathbf{X}^*$.

$\mathbf{X}$ (treated as predictor variable) to each individual dimension of $\mathbf{X}^*$ (treated as response variable). A similar approach was used by [11]. The default number of trees is 500 and the minimum number of samples to split a node $\sqrt{n}$. If a reduced representation $\mathbf{X}^{'} \epsilon \mathbb{R}^{n' \times s}$ is used in Eq. (9), then n' independent regression models are learned that map $\mathbf{X} \epsilon \mathbb{R}^{n \times s}$ to each dimension of $\mathbf{X}^* \epsilon \mathbb{R}^{n' \times s}$. Given a test sample $\mathbf{x}_i$, its representation in the target space can be obtained as $\mathcal{L}(\mathbf{x}_i)$.

D. Discussion

In this section, we highlight the difference of the proposed approach with existing methods and motivate key decisions when learning the target representations.

Comparison to existing methods: Existing methods define an objective function and a set of constraints. The optimization problem usually has one model parameter, the distance metric. Some problems are hard to solve in this manner. The proposed approach can address this limitation - the model parameter is one data point each time. When we solve for a single sample and fix the rest, most of the existing objective functions become convex (i.e., they rely on a combination of convex functions). The target representations learned minimize the objective function, at least locally. We empirically found this solution to be stable. The relationship between the input and the target representations is non-linear. Hence, existing non-linear regression methods can be employed to learn a non-linear distance metric. In our small-scale experiments, we obtained the best performance using generalized regression neural networks. The high computational resources required prohibited a large scale evaluation. Hence, we opted for a naive approach that learns one regression model for each dimension of the target representations (see Sec. 3.4). This approach is not expected to yield optimal performance because it ignores potential correlations of the target representations (i.e., response variables). However, it can be used as proof of concept. More sophisticated methods (such as deep learning) are planned to be investigated in future work.

Target Representations: Different objective functions would yield different target representations. For example, the paper by Globerson and Roweis seeks a projection that collapses all samples with the same label to a single point [5]. Our intuition is that this constraint would restrict the set of feasible solutions and reduce our ability to seek a distance metric with good generalization properties. For instance, it appears easier to find a transformation that maps the input to the target in Fig. I(a) instead to two points, one for each class. Therefore, many researchers propose objective functions that rely on local relationships in the data (e.g., LMNN [14]), a strategy that was also followed in our paper. Concerning the proposed geometric construction, it might be possible to find cases where this approach would not converge to a desired solution. In our experience, though, these cases are rare. By iterating over all samples multiple times, we observed that the same (or approximately same) solution is achieved. A detailed empirical analysis is offered in the appendix.

Hinge Loss: For a given $\mathbf{x}_i$, the LMNN objective function [14] defines the impostors in relation to each target neighbor. That is, $\mathbf{x}_j$ may be an impostor with respect to a target neighbor but not with respect to another. Hence, the hinge loss $\mathcal{H}(\mathbf{x}_i)$ is computed as the sum of all combinations of target neighbors and impostors. This strategy may yield improved robustness in certain cases. To the best of our knowledge, though, it increases the time complexity without yielding significant performance gains in practice. Eq. (3) addresses this problem because the impostors are the same with respect to all target neighbors. As a result, the computational complexity is decreased. Since the number of target neighbors k is usually small, we do not expect outliers to affect the robustness when determining which data points are impostors. The LMNN objective function sets the margin to one for all samples. The reasoning of the authors is that the margin affects the scale of the projection and not the performance of the k-nearest neighbor classifier. In this paper, we minimize the objective function $\mathcal{C}_v(\mathbf{x}_i)$ with the goal of learning a new representation $\mathbf{x}_i^*$. Our intuition is that different areas of the space exhibit different properties (e.g., scaling and class-separation). Hence, the margin should take into consideration these properties and treat them accordingly. The definition of the margin in Eq. (3) addresses this problem. Specifically, the margin (and thus the hinge loss in Eq. (4)) is proportional to the compactness of the target neighbors area and the local scaling of the data.

IV. Experimental Evaluation

In this section, we evaluate different aspects of RBML. First, we assess the performance of RBML in relation to LMNN and GB-LMNN that optimize a similar objective function. Then, we provide comparative results with one of the latest metric learning methods. A comprehensive empirical analysis of RBML is offered in the appendix. Specifically, we study: (i) the convergence properties of RBML-NM and RBML-GE, (ii) the sensitivity of RBML-GE with respect to the parameters α and β, and (iii) the sensitivity of RBML-GE with respect to the output dimensionality and the number of trees. For simplicity, the term RBML is used instead of RBML-GE. The number of nearest neighbors to train LMNN and GB-LMNN is set to three, while the values for all other parameters are cross-validated using the corresponding option provided by the authors. LMNN is initialized using the Principal Component

TABLE I
SUMMARY OF RESULTS FOR EXPERIMENT 1. BOLD FONT IS USED TO DENOTE THE BEST PERFORMANCE.

Method	YTF: CSLBP		YTF: FPLBP		YTF: LBP		DSLR	Webcam	CalTech	Amazon
	ACC	AUC	ACC	AUC	ACC	AUC	Nearest Neighbor Classification Accuracy			
LMNN	68.26	76.86	68.26	76.86	73.40	81.71	72.78	85.08	42.37	**64.74**
GB-LMNN	63.47	71.74	65.20	73.14	70.30	79.57	75.00	**86.67**	44.47	64.33
RBML	**70.46**	**78.11**	**69.30**	**76.90**	**75.02**	**83.57**	**75.56**	80.32	43.86	59.38

Analysis (PCA) projection and GB-LMNN is initialized using the LMNN projection matrix L_0. Unless otherwise indicated, RBML is trained using the defaults parameter values.

A. Experiment 1

In this experiment, we assess the performance of RBML with respect to LMNN and GB-LMNN. All three methods minimize a similar objective function in different ways. LMNN learns a linear projection while GB-LMNN and RBML learn non-linear projections. If RBML outperforms LMNN and matches or surpasses the performance of GB-LMNN, we can conclude that the proposed approach meets the goal of learning a non-linear distance metric. Specifically, we use five publicly available datasets. The YouTube Face (YTF) database is selected because it is designed to facilitate research for the challenging task of video based face verification [15]. YTF contains $3,425$ videos captured from $1,595$ subjects. Following the provided protocol, the samples are divided into ten folds where each contains 250 pairs of samples obtained form the same subject and 250 pairs of samples that belong to different subjects. Each time, the samples of the nine folds are used for training and the samples in the remaining fold are used for testing. YTF comes with three feature descriptors: Local Binary Patterns (LBP), Center-Symmetric LBP (CSLBP), and Four-Patch LBP (FPLBP). The mean vector of each feature is used to represent each video stream. The other four datasets are selected following the paper by Kedem *et al.* [6]. Specifically, the DSLR, Webcam, Amazon, and Caltech datasets comprise 800-bin histograms of visual codebook entries with 157, 295, 958, and 1123 samples, respectively [12]. To compute the classification performance, 80% of the samples were randomly used for training and 20% for testing. This process was repeated 5 times. To reduce the noise and avoid over-fitting, PCA is applied to all the features for all datasets. Specifically, the PCA basis is learned using the training set and the feature length after the projection is set to 100. The training set is used to estimate the mean and standard deviation values for each feature. These values are then used to apply z-score normalization on the raw features of both the training and test sets. For the first four datasets, the 3-nearest neighbor classification performance is reported. For YTF, the accuracy (ACC) and the area under curve (AUC) of the corresponding receiver operating characteristic curve (ROC) are used. The ACC is computed using a threshold that maximizes the accuracy on the training set. To ensure a fair comparison between GB-LMNN and RBML, we initialize both methods using the L_0 projection learned by LMNN. That is, all features for RBML are projected using L_0 before we employ Algorithm 1. We have empirically observed that reducing the output dimensionality for RBML results in improved classification performance. Hence, for YTF, we used Local Fisher Discriminant Analysis (LFDA) [13] to get $\mathbf{X}^{'}$ such that the output dimensionality n' equals 90, 70, and 50 to learn the target representations (see section 3.3). LFDA is selected due to its low computational overhead. To assess the sensitivity of RBML, we compute the performance using different number of trees to train the regression forests. For space, we include the results on the appendix and report only the results for 500 trees (default value) and $n' = 50$. A summary of the results is offered by Table I. For YTF, GB-LMNN appears to result in degraded performance compared to LMNN in all cases. On the contrary, RBML yields increased accuracy and area under the curve for all features. RBML yields the best accuracy for DSLR and improves upon the performance of LMNN for CalTech. However, it appears to degrade the accuracy for the Webcam and Amazon. In five out of seven cases, RBML improved upon LMNN and in four out of seven cases it even surpassed the accuracy obtained by GB-LMNN. Hence, we conclude that RBML can learn a non-linear distance metric.

TABLE III
OVERVIEW OF THE DATASETS USED FOR EXPERIMENT 2.

Dataset	Samples	Dim.	Classes
Iris	150	4	3
Wine	178	13	3
Sonar	208	38	2
Vowel	528	10	11
Balance	625	4	3
Pima	768	8	2
Segm.	2,310	19	7
Letter	20,000	16	26

B. Experiment 2

In this experiment, we duplicate the protocol of [3] to compare the performance of RBML with the Large Margin Nearest Neighbor and Cayley-Klein Metric Learning methods and investigate its beyond recognition tasks. To this end, the Iris, Wine, Sonar, Vowel, Balance, Pima, Segmentation, and Letters datasets from the UCI Machine Learning Repository [1] are used. A summary of key attributes for these datasets is offered by Table III. In all cases, the training set was used to estimate the mean and standard deviation values for each feature. These values were then used to apply z-score

TABLE II
SUMMARY OF RESULTS FOR EXPERIMENT 2. THE VALUES REPORTED ARE IN THE FORMAT: MEAN ACCURACY (STANDARD DEVIATION).

This paper								
Method	Iris	Wine	Sonar	Vowel	Balance	Pima	Segm.	Letters
Euclidean	94.7	95.5	78.9	80.5 (3.7)	81.7 (1.9)	71.3 (1.5)	95.3 (2.0)	95.79 (0.30)
LMNN	**96.0**	**98.3**	83.2	82.7 (2.5)	88.3 (2.0)	71.4 (1.6)	96.8 (1.3)	**96.83** (0.18)
RBML	93.3	**98.3**	**85.1**	**83.0 (3.1)**	**89.7 (1.3)**	**73.4 (2.0)**	**96.9 (1.6)**	96.58 (0.57)
Bi *et al.* [3]								
Method	Iris	Wine	Sonar	Vowel	Balance	Pima	Segm.	Letters
Euclidean	90.0	74.2	77.4	85.0 (3.5)	80.3 (1.9)	70.4 (2.3)	95.3 (2.6)	93.6 (2.5)
LMNN	96.6	95.4	86.9	95.0 (1.7)	87.7 (1.3)	74.8 (1.3)	97.4 (0.9)	97.0 (0.8)
CK-LMNN	**97.0**	**95.5**	**87.1**	**96.1 (1.8)**	**87.9 (1.4)**	**75.2 (1.3)**	**99.7 (0.9)**	**99.8** (0.9)

normalization on the raw features of both the training and test sets. For all datasets, the training set was used to train the LMNN distance metric to obtain the projection matrix L_0, which was then used as an initialization for both CK and RBML. Specifically, the inverse covariance matrix $\mathbf{\Sigma}$ in Eq. (11) of [3] was initialized using $L_0^\top L_0$ to obtain CK-LMNN, and all features were projected using L_0 before employing Algorithm 1 for RBML. The first three datasets (i.e., Iris, Wine, and Sonar) are relatively small. Hence, a leave-one-out cross-validation approach was used to compute the 3-nearest neighbor classification performance for the Euclidean, LMNN, CK-LMNN, and RBML methods. For the Vowel, Balance and Pima datasets, 250 samples were randomly selected as a training set and the rest were used to define the test set. Hence, 278, 375, and 518 test samples were available for each dataset, respectively. This process was repeated 10 times independently. For each dataset and each method, the average accuracy and the corresponding standard deviation values were computed. Finally, for the Segmentation and Letters datasets, a 10-fold cross-validation scheme was used. That is, each dataset was split to 10 random sets of equal size. Nine of the sets were used for training and the remaining set was used to compute the accuracy. This process was repeated 10 times and the average accuracy and the corresponding standard deviation values were computed. A summary of the obtained results is provided by Table II. We were not able to duplicate the accuracy for the Iris, Wine, and Sonar datasets. Also, we obtained inconsistent results for some of the other datasets. For completeness, Table II reports the results from both [3] and our implementation. As illustrated, RBML yields increased accuracy compared with LMNN in five out of seven cases, providing further evidence of the effectiveness of RBML.

V. CONCLUSION

In this paper, we proposed an approach that illustrates how to formulate distance metric learning as a regression problem. First, our method minimizes the objective function to learn target representations. Empirical results illustrate that the proposed geometric construction can achieve this goal more effectively and efficiently compared with a simplex-based method. Then, a distance metric was learned by employing a naive regression method to map the input to the target representations. Two methods that rely on a similar objective function were used as baselines; one of them learns a linear distance metric and the other a non-linear one. By using the linear method as an initialization and solving the objective function using non-linear regression, RBML matched or outperformed the performance of both baselines in most occasions. Hence, we have evidence that RBML can be used to learn non-linear distance metrics with discriminative properties. The sensitivity analyses indicate that RBML is robust to the parameter values. In future work, we plan to investigate whether our approach can produce state-of-the-art results by (i) utilizing more sophisticated regression methods, and (ii) exploring different objective functions.

REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. Technical Report arXiv:1306.6709v3, University of Southern California, Los Angeles, CA 90089, USA, August 19 2013.

[3] Y. Bi, B. Fan, and F. Wu. Beyond mahalanobis metric: cayley-klein metric learning. In *Proc. CVPR*, pages 2339–2347, Boston, MA, June 7 - 12 2015.

[4] L. Breiman. Random forests. *ML*, 45(1):5–32, 2001.

[5] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Proc. NIPS*, pages 451–458, Vancouver, Canada, December 5-8 2005.

[6] D. Kedem, S. Tyree, K. Weinberger, F. Sha, and G. Lanckriet. Non-linear metric learning. In *Proc. NIPS*, pages 2582–2590, Lake Tahoe, Nevada, December 36 2012.

[7] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM*, 9(1):112–147, 1998.

[8] P. Moutafis, M. Leng, and I. Kakadiaris. An overview and empirical comparison of distance metric learning methods. *TCYB*, PP(99):1–14, 2016 (In Press).

[9] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[10] D. Ramanan and S. Baker. Local distance functions: A taxonomy, new algorithms, and an evaluation. *PAMI*, 33(4):794–806, April 2011.

[11] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In *Proc. CVPR*, pages 1685–1692, Colorado Springs, CO, June 24-27 2014.

[12] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proc. ECCV*, pages 213–226, Crete, Greece, Sept. 5-11 2010. Springer.

[13] M. Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *JMLR*, 8:1027–1061, 2007.

[14] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, pages 207–244, 2009.

[15] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. CVPR*, pages 529–534, Colorado Springs, CO, June 21–23 2011.