Parametric Local Metric Learning for Nearest Neighbor Classification

Jun Wang
Department of Computer Science
University of Geneva
Switzerland
Jun.Wang@unige.ch

Adam Woznica
Department of Computer Science
University of Geneva
Switzerland
Adam.Woznica@unige.ch

Alexandros Kalousis
Department of Business Informatics
University of Applied Sciences
Western Switzerland
Alexandros.Kalousis@hesge.ch

Abstract

We study the problem of learning local metrics for nearest neighbor classification. Most previous works on local metric learning learn a number of local unrelated metrics. While this "independence" approach delivers an increased flexibility its downside is the considerable risk of overfitting. We present a new parametric local metric learning method in which we learn a smooth metric matrix function over the data manifold. Using an approximation error bound of the metric matrix function we learn local metrics as linear combinations of basis metrics defined on anchor points over different regions of the instance space. We constrain the metric matrix function by imposing on the linear combinations manifold regularization which makes the learned metric matrix function vary smoothly along the geodesics of the data manifold. Our metric learning method has excellent performance both in terms of predictive power and scalability. We experimented with several large-scale classification problems, tens of thousands of instances, and compared it with several state of the art metric learning methods, both global and local, as well as to SVM with automatic kernel selection, all of which it outperforms in a significant manner.

1 Introduction

The nearest neighbor (NN) classifier is one of the simplest and most classical non-linear classification algorithms. It is guaranteed to yield an error no worse than twice the Bayes error as the number of instances approaches infinity. With finite learning instances, its performance strongly depends on the use of an appropriate distance measure. Mahalanobis metric learning [4, 15, 9, 10, 17, 14] improves the performance of the NN classifier if used instead of the Euclidean metric. It learns a global distance metric which determines the importance of the different input features and their correlations. However, since the discriminatory power of the input features might vary between different neighborhoods, learning a global metric cannot fit well the distance over the data manifold. Thus a more appropriate way is to learn a metric on each neighborhood and *local metric learning* [8, 3, 15, 7] does exactly that. It increases the expressive power of standard Mahalanobis metric learning by learning a number of local metrics (e.g. one per each instance).

Local metric learning has been shown to be effective for different learning scenarios. One of the first local metric learning works, Discriminant Adaptive Nearest Neighbor classification [8], DANN, learns local metrics by shrinking neighborhoods in directions orthogonal to the local decision boundaries and enlarging the neighborhoods parallel to the boundaries. It learns the local metrics independently with no regularization between them which makes it prone to overfitting. The authors of LMNN-Multiple Metric (LMNN-MM) [15] significantly limited the number of learned metrics and constrained all instances in a given region to share the same metric in an effort to combat overfitting. In the supervised setting they fixed the number of metrics to the number of classes; a similar idea has been also considered in [3]. However, they too learn the metrics independently for each region making them also prone to overfitting since the local metrics will be overly specific to their respective regions. The authors of [16] learn local metrics using a least-squares approach by minimizing a weighted sum of the distances of each instance to apriori defined target positions and constraining the instances in the projected space to preserve the original geometric structure of the data in an effort to alleviate overfitting. However, the method learns the local metrics using a learning-order-sensitive propagation strategy, and depends heavily on the appropriate definition of the target positions for each instance, a task far from obvious. In another effort to overcome the overfitting problem of the discriminative methods [8, 15], Generative Local Metric Learning, GLML, [11], propose to learn local metrics by minimizing the NN expected classification error under strong model assumptions. They use the Gaussian distribution to model the learning instances of each class. However, the strong model assumptions might easily be very inflexible for many learning problems.

In this paper we propose the Parametric Local Metric Learning method (PLML) which learns a *smooth metric matrix function* over the data manifold. More precisely, we parametrize the metric matrix of each instance as a linear combination of basis metric matrices of a small set of anchor points; this parametrization is naturally derived from an error bound on local metric approximation. Additionally we incorporate a manifold regularization on the linear combinations, forcing the linear combinations to vary smoothly over the data manifold. We develop an efficient two stage algorithm that first learns the linear combinations of each instance and then the metric matrices of the anchor points. To improve scalability and efficiency we employ a fast first-order optimization algorithm, FISTA [2], to learn the linear combinations as well as the basis metrics of the anchor points. We experiment with the PLML method on a number of large scale classification problems with tens of thousands of learning instances. The experimental results clearly demonstrate that PLML significantly improves the predictive performance over the current state-of-the-art metric learning methods, as well as over multi-class SVM with automatic kernel selection.

2 Preliminaries

We denote by $\mathbf{X}$ the $n \times d$ matrix of learning instances, the i-th row of which is the $\mathbf{x}_i^T \in \mathbb{R}^d$ instance, and by $\mathbf{y} = (y_1, \ldots, y_n)^T$, $y_i \in \{1, \ldots, c\}$ the vector of class labels. The squared Mahalanobis distance between two instances in the input space is given by:

$$d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$$

where $\mathbf{M}$ is a PSD metric matrix ($\mathbf{M} \succeq 0$). A linear metric learning method learns a Mahalanobis metric $\mathbf{M}$ by optimizing some cost function under the PSD constraints for $\mathbf{M}$ and a set of additional constraints on the pairwise instance distances. Depending on the actual metric learning method, different kinds of constraints on pairwise distances are used. The most successful ones are the large margin triplet constraints. A triplet constraint denoted by $c(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, indicates that in the projected space induced by $\mathbf{M}$ the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_k$.

Very often a single metric $\mathbf{M}$ can not model adequately the complexity of a given learning problem in which discriminative features vary between different neighborhoods. To address this limitation in local metric learning we learn a set of local metrics. In most cases we learn a local metric for each learning instance [8, 11], however we can also learn a local metric for some part of the instance space in which case the number of learned metrics can be considerably smaller than n, e.g. [15]. We follow the former approach and learn one local metric per instance. In principle, distances should then be defined as geodesic distances using the local metric on a Riemannian manifold. However, this is computationally difficult, thus we define the distance between instances $\mathbf{x}_i$ and $\mathbf{x}_j$ as:

$$d_{\mathbf{M}_i}^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}_i(\mathbf{x}_i - \mathbf{x}_j)$$

where $\mathbf{M}_i$ is the local metric of instance $\mathbf{x}_i$. Note that most often the local metric $\mathbf{M}_i$ of instance $\mathbf{x}_i$ is different from that of $\mathbf{x}_j$. As a result, the distance $d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j)$ does not satisfy the symmetric property, i.e. it is not a proper metric. Nevertheless, in accordance to the standard practice we will continue to use the term local metric learning following [15, 11].

3 Parametric Local Metric Learning

We assume that there exists a Lipschitz smooth vector-valued function $f(\mathbf{x})$, the output of which is the vectorized local metric matrix of instance $\mathbf{x}$. Learning the local metric of each instance is essentially learning the value of this function at different points over the data manifold. In order to significantly reduce the computational complexity we will approximate the metric function instead of directly learning it.

Definition 1 *A vector-valued function $f(\mathbf{x})$ on $\mathbb{R}^d$ is a (α, β, p)-Lipschitz smooth function with respect to a vector norm $\|\cdot\|$ if $\|f(\mathbf{x}) - f(\mathbf{x}')\| \leq \alpha \|\mathbf{x} - \mathbf{x}'\|$ and $\|f(\mathbf{x}) - f(\mathbf{x}') - \nabla f(\mathbf{x}')^T(\mathbf{x} - \mathbf{x}')\| \leq \beta \|\mathbf{x} - \mathbf{x}'\|^{1+p}$, where $\nabla f(\mathbf{x}')^T$ is the derivative of the f function at $\mathbf{x}'$. We assume $\alpha, \beta > 0$ and $p \in (0, 1]$.*

[18] have shown that any Lipschitz smooth real function $f(\mathbf{x})$ defined on a lower dimensional manifold can be approximated by a linear combination of function values $f(\mathbf{u}), \mathbf{u} \in \mathbf{U}$, of a set $\mathbf{U}$ of anchor points. Based on this result we have the following lemma that gives the respective error bound for learning a Lipschitz smooth vector-valued function.

Lemma 1 *Let $(\boldsymbol{\gamma}, \mathbf{U})$ be a nonnegative weighting on anchor points $\mathbf{U}$ in $\mathbb{R}^d$. Let f be an (α, β, p)-Lipschitz smooth vector function. We have for all $\mathbf{x} \in \mathbb{R}^d$:*

$$\left\| f(\mathbf{x}) - \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) f(\mathbf{u}) \right\| \leq \alpha \left\| \mathbf{x} - \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) \mathbf{u} \right\| + \beta \sum_{\mathbf{u} \in \mathbf{U}} \gamma_{\mathbf{u}}(\mathbf{x}) \|\mathbf{x} - \mathbf{u}\|^{1+p} \quad (1)$$

The proof of the above Lemma 1 is similar to the proof of Lemma 2.1 in [18]; for lack of space we omit its presentation. By the nonnegative weighting strategy $(\boldsymbol{\gamma}, \mathbf{U})$, the PSD constraints on the approximated local metric is automatically satisfied if the local metrics of anchor points are PSD matrices.

Lemma 1 suggests a natural way to approximate the local metric function by parameterizing the metric $\mathbf{M}_i$ of each instance $\mathbf{x}_i$ as a *weighted linear combination*, $\mathbf{W}_i \in \mathbb{R}^m$, of a small set of metric basis, $\{\mathbf{M}_{b_1}, \dots, \mathbf{M}_{b_m}\}$, each one associated with an anchor point defined in some region of the instance space. This parametrization will also provide us with a global way to regularize the flexibility of the metric function. We will first learn the vector of weights $\mathbf{W}_i$ for each instance $\mathbf{x}_i$, and then the basis metric matrices; these two together, will give us the $\mathbf{M}_i$ metric for the instance $\mathbf{x}_i$.

More formally, we define a $m \times d$ matrix $\mathbf{U}$ of anchor points, the i-th row of which is the anchor point $\mathbf{u}_i$, where $\mathbf{u}_i^T \in \mathbb{R}^d$. We denote by $\mathbf{M}_{b_i}$ the Mahalanobis metric matrix associated with $\mathbf{u}_i$. The anchor points can be defined using some clustering algorithm, we have chosen to define them as the means of clusters constructed by the k-means algorithm. The local metric $\mathbf{M}_i$ of an instance $\mathbf{x}_i$ is parametrized by:

$$\mathbf{M}_i = \sum_{b_k} W_{ib_k} \mathbf{M}_{b_k}, \ W_{ib_k} \geq 0, \ \sum_{b_k} W_{ib_k} = 1 \quad (2)$$

where $\mathbf{W}$ is a $n \times m$ weight matrix, and its W_{ib_k} entry is the weight of the basis metric $\mathbf{M}_{b_k}$ for the instance $\mathbf{x}_i$. The constraint $\sum_{b_k} W_{ib_k} = 1$ removes the scaling problem between different local metrics. Using the parametrization of equation (2), the squared distance of $\mathbf{x}_i$ to $\mathbf{x}_j$ under the metric $\mathbf{M}_i$ is:

$$d^2_{\mathbf{M}_i}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{b_k} W_{ib_k} d^2_{\mathbf{M}_{b_k}}(\mathbf{x}_i, \mathbf{x}_j) \quad (3)$$

where $d^2_{\mathbf{M}_{b_k}}(\mathbf{x}_i, \mathbf{x}_j)$ is the squared Mahalanobis distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ under the basis metric $\mathbf{M}_{b_k}$. We will show in the next section how to learn the weights of the basis metrics for each instance and in section 3.2 how to learn the basis metrics.

Algorithm 1 Smoothl Local Linear Weight Learning

Input: $\mathbf{W^0}, \mathbf{X}, \mathbf{U}, \mathbf{G}, \mathbf{L}, \lambda_1$, and λ_2
Output: matrix $\mathbf{W}$
define $\widetilde{g}_{\beta,\mathbf{Y}}(\mathbf{W}) = g(\mathbf{Y}) + tr(\nabla g(\mathbf{Y})^T(\mathbf{W} - \mathbf{Y})) + \frac{\beta}{2}\|\mathbf{W} - \mathbf{Y}\|_F^2$
initialize: $t_1 = 1, \beta = 1, \mathbf{Y}^1 = \mathbf{W^0}$, and $i = 0$
repeat
   $i = i + 1, \mathbf{W}^i = Proj((\mathbf{Y}^i - \frac{1}{\beta}\nabla g(\mathbf{Y}^i)))$
   while $g(\mathbf{W}^i) > \widetilde{g}_{\beta,\mathbf{Y}^i}(\mathbf{W}^i)$ **do**
     $\beta = 2\beta, \mathbf{W}^i = Proj((\mathbf{Y}^i - \frac{1}{\beta}\nabla g(\mathbf{Y}^i)))$
   end while
   $t_{i+1} = \frac{1+\sqrt{1+4t_i^2}}{2}, \mathbf{Y}^{i+1} = \mathbf{W}^i + \frac{t_i-1}{t_{i+1}}(\mathbf{W}^i - \mathbf{W}^{i-1})$
until converges;

3.1 Smooth Local Linear Weighting

Lemma 1 bounds the approximation error by two terms. The first term states that $\mathbf{x}$ should be close to its linear approximation, and the second that the weighting should be local. In addition we want the local metrics to vary smoothly over the data manifold. To achieve this smoothness we rely on manifold regularization and constrain the weight vectors of neighboring instances to be similar. Following this reasoning we will learn Smooth Local Linear Weights for the basis metrics by minimizing the error bound of (1) together with a regularization term that controls the weight variation of similar instances. To simplify the objective function, we use the term $\left\|\mathbf{x} - \sum_{\mathbf{u}\in\mathbf{U}}\gamma_{\mathbf{u}}(\mathbf{x})\mathbf{u}\right\|^2$ instead of $\left\|\mathbf{x} - \sum_{\mathbf{u}\in\mathbf{U}}\gamma_{\mathbf{u}}(\mathbf{x})\mathbf{u}\right\|$. By including the constraints on the $\mathbf{W}$ weight matrix in (2), the optimization problem is given by:

$$\min_{\mathbf{W}} g(\mathbf{W}) = \|\mathbf{X} - \mathbf{WU}\|_F^2 + \lambda_1 tr(\mathbf{WG}) + \lambda_2 tr(\mathbf{W^T LW}) \tag{4}$$

$$s.t. \quad \mathbf{W}_{ib_k} \geq 0, \sum_{b_k}\mathbf{W}_{ib_k} = 1, \forall i, b_k$$

where $tr(\cdot)$ and $\|\cdot\|_F$ denote respectively the trace norm of a square matrix and the Frobenius norm of a matrix. The $m \times n$ matrix $\mathbf{G}$ is the squared distance matrix between each anchor point $\mathbf{u}_i$ and each instance $\mathbf{x}_j$, obtained for $p = 1$ in (1), i.e. its (i,j) entry is the squared Euclidean distance between $\mathbf{u}_i$ and $\mathbf{x}_j$. $\mathbf{L}$ is the $n \times n$ Laplacian matrix constructed by $\mathbf{D} - \mathbf{S}$, where $\mathbf{S}$ is the $n \times n$ symmetric pairwise similarity matrix of learning instances and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_k \mathbf{S}_{ik}$. Thus the minimization of the $tr(\mathbf{W^T LW})$ term constrains similar instances to have similar weight coefficients. The minimization of the $tr(\mathbf{WG})$ term forces the weights of the instances to reflect their local properties. Most often the similarity matrix $\mathbf{S}$ is constructed using k-nearest neighbors graph [19]. The λ_1 and λ_2 parameters control the importance of the different terms.

Since the cost function $g(\mathbf{W})$ is convex quadratic with $\mathbf{W}$ and the constraint is simply linear, (4) is a convex optimization problem with a unique optimal solution. The constraints on $\mathbf{W}$ in (4) can be seen as n simplex constraints on each row of $\mathbf{W}$; we will use the projected gradient method to solve the optimization problem. At each iteration t, the learned weight matrix $\mathbf{W}$ is updated by:

$$\mathbf{W}^{t+1} = Proj(\mathbf{W}^t - \eta\nabla g(\mathbf{W}^t)) \tag{5}$$

where $\eta > 0$ is the step size and $\nabla g(\mathbf{W}^t)$ is the gradient of the cost function $g(\mathbf{W})$ at $\mathbf{W}^t$. The $Proj(\cdot)$ denotes the simplex projection operator on each row of $\mathbf{W}$. Such a projection operator can be efficiently implemented with a complexity of $O(nm\log(m))$ [6]. To speed up the optimization procedure we employ a fast first-order optimization method FISTA, [2]. The detailed algorithm is described in Algorithm 1. The Lipschitz constant β required by this algorithm is estimated by using the condition of $g(\mathbf{W}^i) \leq \widetilde{g}_{\beta,\mathbf{Y}^i}(\mathbf{W}^i)$ [1]. At each iteration, the main computations are in the gradient and the objective value with complexity $O(nmd + n^2m)$.

To set the weights of the basis metrics for a testing instance we can optimize (4) given the weight of the basis metrics for the training instances. Alternatively we can simply set them as the weights of its nearest neighbor in the training instances. In the experiments we used the latter approach.

3.2 Large Margin Basis Metric Learning

In this section we define a large margin based algorithm to learn the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$. Given the $\mathbf{W}$ weight matrix of basis metrics obtained using Algorithm 1, the local metric $\mathbf{M}_i$ of an instance $\mathbf{x}_i$ defined in (2) is linear with respect to the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$. We define the relative comparison distance of instances $\mathbf{x}_i$, $\mathbf{x}_j$ and $\mathbf{x}_k$ as: $d^2_{\mathbf{M}_i}(\boldsymbol{x}_i, \boldsymbol{x}_k) - d^2_{\mathbf{M}_i}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In a large margin constraint $c(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$, the squared distance $d^2_{\mathbf{M}_i}(\boldsymbol{x}_i, \boldsymbol{x}_k)$ is required to be larger than $d^2_{\mathbf{M}_i}(\boldsymbol{x}_i, \boldsymbol{x}_j) + 1$, otherwise an error $\boldsymbol{\xi}_{ijk} \geq 0$ is generated. Note that, this relative comparison definition is different from that defined in LMNN-MM [15]. In LMNN-MM to avoid over-fitting, different local metrics $\mathbf{M}_j$ and $\mathbf{M}_k$ are used to compute the squared distance $d^2_{\mathbf{M}_j}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $d^2_{\mathbf{M}_k}(\boldsymbol{x}_i, \boldsymbol{x}_k)$ respectively, as no smoothness constraint is added between metrics of different local regions.

Given a set of triplet constraints, we learn the basis metrics $\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}$ with the following optimization problem:

$$\min_{\mathbf{M}_{b_1}, \ldots, \mathbf{M}_{b_m}, \boldsymbol{\xi}} \quad \alpha_1 \sum_{b_l} ||\mathbf{M}_{b_l}||^2_F + \sum_{ijk} \boldsymbol{\xi}_{ijk} + \alpha_2 \sum_{ij} \sum_{b_l} W_{ib_l} d^2_{\mathbf{M}_{b_l}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad (6)$$

$$s.t. \quad \sum_{b_l} W_{ib_l}(d^2_{\mathbf{M}_{b_l}}(\boldsymbol{x}_i, \boldsymbol{x}_k) - d^2_{\mathbf{M}_{b_l}}(\boldsymbol{x}_i, \boldsymbol{x}_j)) \geq 1 - \boldsymbol{\xi}_{ijk} \; \forall i, j, k$$

$$\boldsymbol{\xi}_{ijk} \geq 0; \; \forall i, j, k \; \mathbf{M}_{b_l} \succeq 0; \; \forall b_l$$

where α_1 and α_2 are parameters that balance the importance of the different terms. The large margin triplet constraints for each instance are generated using its k_1 same class nearest neighbors and k_2 different class nearest neighbors by requiring its distances to the k_2 different class instances to be larger than those to its k_1 same class instances. In the objective function of (6) the basis metrics are learned by minimizing the sum of large margin errors and the sum of squared pairwise distances of each instance to its k_1 nearest neighbors computed using the local metric. Unlike LMNN we add the squared Frobenius norm on each basis metrics in the objective function. We do this for two reasons. First we exploit the connection between LMNN and SVM shown in [5] under which the squared Frobenius norm of the metric matrix is related to the SVM margin. Second because adding this term leads to an easy-to-optimize dual formulation of (6) [12].

Unlike many special solvers which optimize the primal form of the metric learning problem [15, 13], we follow [12] and optimize the Lagrangian dual problem of (6). The dual formulation leads to an efficient basis metric learning algorithm. Introducing the Lagrangian dual multipliers $\boldsymbol{\gamma}_{ijk}$, $\boldsymbol{p}_{ijk}$ and the PSD matrices $\mathbf{Z}_{b_l}$ to respectively associate with every large margin triplet constraints, $\boldsymbol{\xi}_{ijk} \geq 0$ and the PSD constraints $\mathbf{M}_{b_l} \succeq 0$ in (6), we can easily derive the following Lagrangian dual form

$$\max_{\mathbf{Z}_{b_1}, \ldots, \mathbf{Z}_{b_m}, \boldsymbol{\gamma}} \quad \sum_{ijk} \gamma_{ijk} - \sum_{b_l} \frac{1}{4\alpha_1} \cdot \|\mathbf{Z}_{b_l} + \sum_{ijk} \gamma_{ijk} W_{ib_l} \mathbf{C}_{ijk} - \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij}\|^2_F \qquad (7)$$

$$s.t. \quad 1 \geq \gamma_{ijk} \geq 0; \; \forall_{i,j,k} \; \mathbf{Z}_{b_l} \succeq 0; \; \forall b_l$$

and the corresponding optimality conditions: $\mathbf{M}^*_{b_l} = \frac{(\mathbf{Z}^*_{b_l} + \sum_{ijk} \gamma^*_{ijk} W_{ib_l} \mathbf{C}_{ijk} - \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij})}{2\alpha_1}$ and $1 \geq \gamma_{ijk} \geq 0$, where the matrices $\mathbf{A}_{ij}$ and $\mathbf{C}_{ijk}$ are given by $\mathbf{x}_{ij}^T \mathbf{x}_{ij}$ and $\mathbf{x}_{ik}^T \mathbf{x}_{ik} - \mathbf{x}_{ij}^T \mathbf{x}_{ij}$ respectively, where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

Compared to the primal form, the main advantage of the dual formulation is that the second term in the objective function of (7) has a closed-form solution for $\mathbf{Z}_{b_l}$ given a fixed $\boldsymbol{\gamma}$. To drive the optimal solution of $\mathbf{Z}_{b_l}$, let $\mathbf{K}_{b_l} = \alpha_2 \sum_{ij} W_{ib_l} \mathbf{A}_{ij} - \sum_{ijk} \gamma_{ijk} W_{ib_l} \mathbf{C}_{ijk}$. Then, given a fixed $\boldsymbol{\gamma}$, the optimal solution of $\mathbf{Z}_{b_l}$ is $\mathbf{Z}^*_{b_l} = (\mathbf{K}_{b_l})_+$, where $(\mathbf{K}_{b_l})_+$ projects the matrix $\mathbf{K}_{b_l}$ onto the PSD cone, i.e. $(\mathbf{K}_{b_l})_+ = \mathbf{U}[\max(\mathbf{diag}(\boldsymbol{\Sigma})), \mathbf{0})]\mathbf{U^T}$ with $\mathbf{K}_{b_l} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U^T}$.

Now, (7) is rewritten as:

$$\min_{\boldsymbol{\gamma}} \quad g(\boldsymbol{\gamma}) = -\sum_{ijk} \gamma_{ijk} + \sum_{b_l} \frac{1}{4\alpha_1} \|(\mathbf{K}_{b_l})_+ - \mathbf{K}_{b_l}\|^2_F \qquad (8)$$

$$s.t. \quad 1 \geq \gamma_{ijk} \geq 0; \forall i, j, k$$

And the optimal condition for $\mathbf{M}_{b_l}$ is $\mathbf{M}_{b_l}^* = \frac{1}{2\alpha_1}((\mathbf{K}_{b_l}^*)_+ - \mathbf{K}_{b_l}^*)$. The gradient of the objective function in (8), $\nabla g(\gamma_{ijk})$, is given by: $\nabla g(\gamma_{ijk}) = -1 + \sum_{b_l} \frac{1}{2\alpha_1} \langle (\mathbf{K}_{b_l})_+ - \mathbf{K}_{b_l}, W_{ib_l}\mathbf{C}_{ijk} \rangle$. At each iteration, γ is updated by: $\gamma^{i+1} = BoxProj(\gamma^i - \eta \nabla g(\gamma^i))$ where $\eta > 0$ is the step size. The $BoxProj(\cdot)$ denotes the simple box projection operator on γ as specified in the constraints of (8). At each iteration, the main computational complexity lies in the computation of the eigen-decomposition with a complexity of $O(md^3)$ and the computation of the gradient with a complexity of $O(m(nd^2 + cd))$, where m is the number of basis metrics and c is the number of large margin triplet constraints. As in the weight learning problem the FISTA algorithm is employed to accelerate the optimization process; for lack of space we omit the algorithm presentation.

4   Experiments

In this section we will evaluate the performance of PLML and compare it with a number of relevant baseline methods on six datasets with large number of instances, ranging from 5K to 70K instances; these datasets are Letter, USPS, Pendigits, Optdigits, Isolet and MNIST. We want to determine whether the addition of manifold regularization on the local metrics improves the predictive performance of local metric learning, and whether the local metric learning improves over learning with single global metric. We will compare PLML against six baseline methods. The first, SML, is a variant of PLML where a single global metric is learned, i.e. we set the number of basis in (6) to one. The second, Cluster-Based LML (CBLML), is also a variant of PLML without weight learning. Here we learn one local metric for each cluster and we assign a weight of one for a basis metric $\mathbf{M}_{b_i}$ if the corresponding cluster of $\mathbf{M}_{b_i}$ contains the instance, and zero otherwise. Finally, we also compare against four state of the art metric learning methods LMNN [15], BoostMetric [13][1], GLML [11] and LMNN-MM [15][2]. The former two learn a single global metric and the latter two a number of local metrics. In addition to the different metric learning methods, we also compare PLML against multi-class SVMs in which we use the one-against-all strategy to determine the class label for multi-class problems and select the best kernel with inner cross validation.

Since metric learning is computationally expensive for datasets with large number of features we followed [15] and reduced the dimensionality of the USPS, Isolet and MINIST datasets by applying PCA. In these datasets the retained PCA components explain 95% of their total variances. We preprocessed all datasets by first standardizing the input features, and then normalizing the instances to so that their L2-norm is one.

PLML has a number of hyper-parameters. To reduce the computational time we do not tune λ_1 and λ_2 of the weight learning optimization problem (4), and we set them to their default values of $\lambda_1 = 1$ and $\lambda_2 = 100$. The Laplacian matrix $\mathbf{L}$ is constructed using the six nearest neighbors graph following [19]. The anchor points $\mathbf{U}$ are the means of clusters constructed with k-means clustering. The number m of anchor points, i.e. the number of basis metrics, depends on the complexity of the learning problem. More complex problems will often require a larger number of anchor points to better model the complexity of the data. As the number of classes in the examined datasets is 10 or 26, we simply set $m = 20$ for all datasets. In the basis metric learning problem (6), the number of the dual parameters γ is the same as the number of triplet constraints. To speedup the learning process, the triplet constraints are constructed only using the three same-class and the three different-class nearest neighbors for each learning instance. The parameter α_2 is set to 1, while the parameter α_1 is the only parameter that we select from the set $\{0.01, 0.1, 1, 10, 100\}$ using 2-fold inner cross-validation. The above setting of basis metric learning for PLML is also used with the SML and CBLML methods. For LMNN and LMNN-MM we use their default settings, [15], in which the triplet constraints are constructed by the three nearest same-class neighbors and all different-class samples. As a result, the number of triplet constraints optimized in LMNN and LMNN-MM is much larger than those of PLML, SML, BoostMetric and CBLML. The local metrics are initialized by identity matrices. As in [11], GLML uses the Gaussian distribution to model the learning instances from the same class. Finally, we use the 1-NN rule to evaluate the performance of the different metric learning methods. In addition as we already mentioned we also compare against multi-class SVM. Since the performance of the latter depends heavily on the kernel with which it is coupled we do automatic kernel selection with inner cross validation to select the best

---

[1]http://code.google.com/p/boosting
[2]http://www.cse.wustl.edu/~kilian/code/code.html.

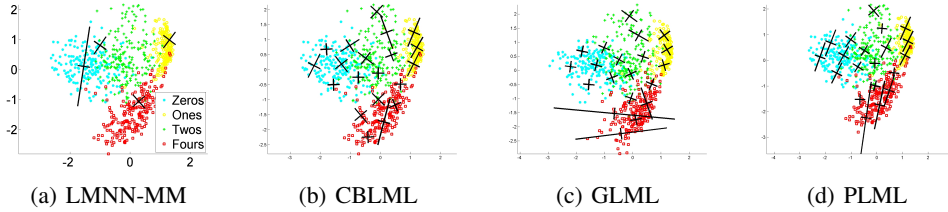| | (a) LMNN-MM | (b) CBLML | (c) GLML | (d) PLML |

Figure 1: The visualization of learned local metrics of LMNN-MM, CBLML, GLML and PLML.

Table 1: Accuracy results. The superscripts $^{+-=}$ next to the accuracies of PLML indicate the result of the McNemar's statistical test with LMNN, BoostMetric, SML, CBLML, LMNN-MM, GMLM and SVM. They denote respectively a significant win, loss or no difference for PLML. The number in the parenthesis indicates the score of the respective algorithm for the given dataset based on the pairwise comparisons of the McNemar's statistical test.

Datasets		PLML		Single Metric Learning Baselines			Local Metric Learning Baselines				SVM
				LMNN	BoostMetric	SML	CBLML	LMNN-MM	GLML		
Letter		$\mathbf{97.22}^{+++|+++|+}$(7.0)		96.08(2.5)	96.49(4.5)	96.71(5.5)	95.82(2.5)	95.02(1.0)	93.86(0.0)		96.64(5.0)
Pendigits		$\mathbf{98.34}^{+++|+++|+}$(7.0)		97.43(2.0)	97.43(2.5)	97.80(4.5)	97.94(5.0)	97.43(2.0)	96.88(0.0)		97.91(5.0)
Optdigits		$\mathbf{97.72}^{===|+++|=}$(5.0)		$\mathbf{97.55}$(5.0)	$\mathbf{97.61}$(5.0)	$\mathbf{97.22}$(5.0)	95.94(1.5)	95.94(1.5)	94.82(0.0)		$\mathbf{97.33}$(5.0)
Isolet		$\mathbf{95.25}^{=+=|+++|=}$(5.5)		$\mathbf{95.51}$(5.5)	89.16(2.5)	$\mathbf{94.68}$(5.5)	89.03(2.5)	84.61(0.5)	84.03(0.5)		$\mathbf{95.19}$(5.5)
USPS		$\mathbf{98.26}^{+++|+++|=}$(6.5)		97.92(4.5)	97.65(2.5)	97.94(4.0)	96.22(0.5)	97.90(4.0)	96.05(0.5)		$\mathbf{98.19}$(5.5)
MNIST		$\mathbf{97.30}^{=++|+++|=}$(6.0)		$\mathbf{97.30}$(6.0)	96.03(2.5)	96.57(4.0)	95.77(2.5)	93.24(1.0)	84.02(0.0)		$\mathbf{97.62}$(6.0)
Total Score		37		25.5	19.5	28.5	14.5	10	1		32.5

kernel and parameter setting. The kernels were chosen from the set of linear, polynomial (degree 2,3 and 4), and Gaussian kernels; the width of the Gaussian kernel was set to the average of all pairwise distances. Its C parameter of the hinge loss term was selected from $\{0.1, 1, 10, 100\}$.

To estimate the classification accuracy for Pendigits, Optdigits, Isolet and MNIST we used the default train and test split, for the other datasets we used 10-fold cross-validation. The statistical significance of the differences were tested with McNemar's test with a p-value of 0.05. In order to get a better understanding of the relative performance of the different algorithms for a given dataset we used a simple ranking schema in which an algorithm A was assigned one point if it was found to have a statistically significantly better accuracy than another algorithm B, 0.5 points if the two algorithms did not have a significant difference, and zero points if A was found to be significantly worse than B.

4.1 Results

In Table 1 we report the experimental results. PLML consistently outperforms the single global metric learning methods LMNN, BoostMetric and SML, for all datasets except Isolet on which its accuracy is slightly lower than that of LMNN. Depending on the single global metric learning method with which we compare it, it is significantly better in three, four, and five datasets (for LMNN, SML, and BoostMetric respectively), out of the six and never significantly worse. When we compare PLML with CBLML and LMNN-MM, the two baseline methods which learn one local metric for each cluster and each class respectively with no smoothness constraints, we see that it is statistically significantly better in all the datasets. GLML fails to learn appropriate metrics on all datasets because its fundamental generative model assumption is often not valid. Finally, we see that PLML is significantly better than SVM in two out of the six datasets and it is never significantly worse; remember here that with SVM we also do inner fold kernel selection to automatically select the appropriate feature space. Overall PLML is the best performing methods scoring 37 points over the different datasets, followed by SVM with automatic kernel selection and SML which score 32.5 and 28.5 points respectively. The other metric learning methods perform rather poorly.

Examining more closely the performance of the baseline local metric learning methods CBLML and LMNN-MM we observe that they tend to overfit the learning problems. This can be seen by their considerably worse performance with respect to that of SML and LMNN which rely on a single global model. On the other hand PLML even though it also learns local metrics it does not suffer from the overfitting problem due to the manifold regularization. The poor performance of LMNN-
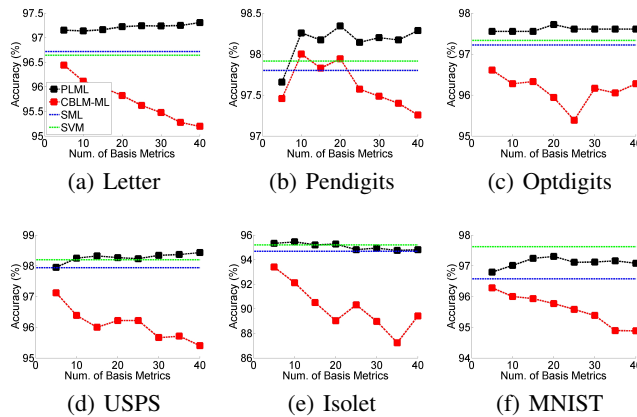
7

Figure 2: Accuracy results of PLML and CBLML with varying number of basis metrics.

MM is not in agreement with the results reported in [15]. The main reason for the difference is the experimental setting. In [15], 30% of the training instance of each dataset were used as a validation set to avoid overfitting.

To provide a better understanding of the behavior of the learned metrics, we applied PLML LMNN-MM, CBLML and GLML, on an image dataset containing instances of four different handwritten digits, zero, one, two, and four, from the MNIST dataset. As in [15], we use the two main principal components to learn. Figure 1 shows the learned local metrics by plotting the axis of their corresponding ellipses(black line). The direction of the longer axis is the more discriminative. Clearly PLML fits the data much better than LMNN-MM and as expected its local metrics vary smoothly. In terms of the predictive performance, PLML has the best with 82.76% accuracy. The CBLML, LMNN-MM and GLML have an almost identical performance with respective accuracies of 82.59%, 82.56% and 82.51%.

Finally we investigated the sensitivity of PLML and CBLML to the number of basis metrics, we experimented with $m \in \{5, 10, 15, 20, 25, 30, 35, 40\}$. The results are given in Figure 2. We see that the predictive performance of PLML often improves as we increase the number of the basis metrics. Its performance saturates when the number of basis metrics becomes sufficient to model the underlying training data. As expected different learning problems require different number of basis metrics. PLML does not overfit on any of the datasets. In contrast, the performance of CBLML gets worse when the number of basis metrics is large which provides further evidence that CBLML does indeed overfit the learning problems, demonstrating clearly the utility of the manifold regularization.

5 Conclusions

Local metric learning provides a more flexible way to learn the distance function. However they are prone to overfitting since the number of parameters they learn can be very large. In this paper we presented PLML, a local metric learning method which regularizes local metrics to vary smoothly over the data manifold. Using an approximation error bound of the metric matrix function, we parametrize the local metrics by a weighted linear combinations of local metrics of anchor points. Our method scales to learning problems with tens of thousands of instances and avoids the overfitting problems that plague the other local metric learning methods. The experimental results show that PLML outperforms significantly the state of the art metric learning methods and it has a performance which is significantly better or equivalent to that of SVM with automatic kernel selection.

References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*.

[2] A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal-recovery problems. *Convex Optimization in Signal Processing and Communications*, pages 42–88, 2010.

[3] M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, page 11, 2004.

[4] J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[5] H. Do, A. Kalousis, J. Wang, and A. Woznica. A metric learning perspective of svm: on the relation of svm and lmnn. *AISTATS*, 2012.

[6] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *ICML*, 2008.

[7] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems*, volume 19, pages 417–424. MIT Press, 2007.

[8] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. on PAMI*, 1996.

[9] P. Jain, B. Kulis, J.V. Davis, and I.S. Dhillon. Metric and kernel learning using a linear transformation. *JMLR*, 2012.

[10] R. Jin, S. Wang, and Y. Zhou. Regularized distance metric learning: Theory and algorithm. In *NIPS*, 2009.

[11] Y.K. Noh, B.T. Zhang, and D.D. Lee. Generative local metric learning for nearest neighbor classification. *NIPS*, 2009.

[12] C. Shen, J. Kim, and L. Wang. A scalable dual approach to semidefinite metric learning. In *CVPR*, 2011.

[13] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning using boosting-like algorithms. *JMLR*, 2012.

[14] J. Wang, H. Do, A. Woznica, and A. Kalousis. Metric learning with multiple kernels. In *NIPS*, 2011.

[15] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009.

[16] D.Y. Yeung and H. Chang. Locally smooth metric learning with application to image retrieval. In *ICCV*, 2007.

[17] Y. Ying, K. Huang, and C. Campbell. Sparse metric learning via smooth optimization. *NIPS*, 2009.

[18] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *NIPS*, 2009.

[19] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *NIPS*, 2004.