
There may be differences between this version and the published version. You are advised to consult the publisher’s version if you wish to cite from it.

http://eprints.gla.ac.uk/225908/

Deposited on: 5 November 2020
Learning Local Metrics and Influential Regions for Classification

Mingzhi Dong, Yujiang Wang, Xiaochen Yang, Jing-Hao Xue

Abstract—The performance of distance-based classifiers heavily depends on the underlying distance metric, so it is valuable to learn a suitable metric from the data. To address the problem of multimodality, it is desirable to learn local metrics. In this short paper, we define a new intuitive distance with local metrics and influential regions, and subsequently propose a novel local metric learning algorithm called LM-LIR for distance-based classification. Our key intuition is to partition the metric space into influential regions and a background region, and then regularize the effectiveness of each local metric to be within the related influential regions. We learn multiple local metrics and influential regions to reduce the empirical hinge loss, and regularize the parameters on the basis of a resultant learning bound. Encouraging experimental results are obtained from various public and popular data sets.

Index Terms—Distance-based classification, distance metric, metric learning, local metric.

1 INTRODUCTION

Classification is a fundamental task in the field of machine learning. While deep learning classifiers have obtained superior performance on numerous applications, they generally require a large amount of labeled data. For small data sets, traditional classification algorithms remain valuable.

The nearest neighbor (NN) classifier is one of the oldest established methods for classification, which compares the distances between a new instance and the training instances. However, with different metrics, the performance of NN would be quite different. Hence it is very beneficial if we can find a well-suited and adaptive distance metric for specific applications. To this end, metric learning is an appealing technique. It enables the algorithms to automatically learn a metric from the available data. Metric learning with a convex objective function was first proposed in the seminal work of Xing et al. [1]. After that, many other metric learning methods have been developed and widely adopted, such as the large margin nearest neighbor (LMNN) [2] and the information theoretic metric learning [3]. Some theoretical work has also been proposed for metric learning, especially on deriving different generalization bounds [4]–[7] and deep networks have been used to represent nonlinear metrics [8], [9]. In addition, metric learning methods have been developed for specific purposes, including multi-output tasks [10], multi-view learning [11], medical image retrieval [12], kinship verification tasks [13], face recognition tasks [14], tracking problems [15] and so on.

Most aforementioned methods use a single metric for the whole metric space and thus may not be well-suited for data sets with multimodality. To solve this problem, local metric learning algorithms have been proposed [2], [16]–[23].

Most of these localized algorithms can be categorized into two groups: 1) Each data point or cluster of data points has a local metric $M(x_i)$. This, however, results in an asymmetric distance as illustrated in [17], i.e. $M(x_i) \neq M(x_j)$ would cause $D(x_i, x_j; M(x_i)) \neq D(x_j, x_i; M(x_j))$. 2) Each line segment or cluster of line segments has a local metric, i.e. $M(x_i, x_j)$. In [19], $M(x_i, x_j) = \sum_k w_k(x_i, x_j)M_k$, where $w_k$ is defined as $P(k|x_i) + P(k|x_j)$ so as to guarantee the symmetry and $P(k|x_i)$ or $P(k|x_j)$ is based on the posterior probability that the point $x$ belongs to the $k$th Gaussian cluster in a Gaussian mixture (GMM). However, most of the line segment approaches are based on certain heuristic design. Geometric properties of line segments, which are very intuitive and interpretable, have scarcely been considered.

In this short paper, we define a geometrically interpretable, symmetric distance, and propose a novel local metric learning algorithm that learns local metrics and locations of the local metrics simultaneously; the proposed method is termed as LM-LIR. By splitting the metric space into influential regions and a background region, we define the distance between any two points as the sum of lengths of line segments in each region, as illustrated in Fig. 1. Building multiple influential regions solves the multimodality issues; and learning a suitable local metric in...
In this section, we will first define influential regions. Each influential region improves class separability, as shown in Fig. 2.

To establish our new distance and local metric learning method, we first define some key concepts, namely influential regions, local metrics and line segments, which lead to the definition of the new distance. Then we calculate the distance by discussing the geometric relationship between line segment and influential regions. After that, we use the proposed local metric to build a novel classifier and study its learnability. The method, we first define some key concepts, namely influential regions, local metrics and line segments, which lead to the definition of the new distance. Then we calculate the distance by discussing the geometric relationship between line segment and influential regions. After that, we use the proposed local metric to build a novel classifier and study its learnability.

2 Definitions of Influential Regions, Local Metrics and Distance

In this section, we will first define influential regions $A_s, s = 1, \ldots, S$, and the background region $B$. With a local metric for each region $M(A_s)$ and $M(B)$, the distance between $x_i$ and $x_j$ will be defined as the sum of lengths of line segments in each influential region and the background region, as illustrated in Fig. 1. Since the metric is defined with respect to line segments, the distance is symmetric, i.e., $D(x_i, x_j) = D_M(x_i, x_j) = D_M(x_j, x_i)$. To simplify the calculation required later, we restrict the shape of each influential region to be a ball.

**Definition 1.** Influential regions are defined to be any set of balls or hyperspheres inside the metric space:

$$A = \{ A_s, s = 1, \ldots, S \},$$

where $S$ denotes the number of influential regions; $A_s = \text{Ball}(o_s, r_s)$, a ball with the center $o_s$ and radius $r_s$. Points $x \in A_s$ construct a set with the following form:

$$\{ x | (o_s - x)^T (o_s - x) \leq r_s^2 \}. \quad (1)$$

The location of each influential region is determined by using the Euclidean distance.

**Definition 2.** Background region is defined to be the region excluding influential regions:

$$B = U - \bigcup_{s=1,\ldots,S} A_s,$$

where $U$ denotes the universe set.

Throughout this paper, the distance between two points $x_i$ and $x_j$ is equivalent to the length of line segment $\overrightarrow{x_i x_j}$, i.e., $D(x_i, x_j) = l(\overrightarrow{x_i x_j})$. Length $l(\overrightarrow{x_i x_j})$ in influential regions and the background region will be defined separately with respective metrics.

**Definition 3.** Each influential region $A_s$ has its own local metric $M(A_s)$. The length of a line segment $\overrightarrow{x_i x_j}$ inside an influential region $A_s$ is defined as

$$l(\overrightarrow{x_i x_j}; M(A_s)) = D_M(A_s)(x_i, x_j) = \sqrt{(x_i - x_j)^T M(A_s) (x_i - x_j)}. \quad (2)$$

Here, we adopt the Mahalanobis distance, rather than the widely used squared Mahalanobis distance, since it simplifies the further optimization problem.

**Definition 4.** The background region $B$ has a background metric $M(B)$. For any two points $x_i, x_j \in B$ and $\overrightarrow{x_i x_j} \subseteq B$, the length of a line segment is defined as

$$l(\overrightarrow{x_i x_j}; M(B)) = D_M(B)(x_i, x_j) = \sqrt{(x_i - x_j)^T M(B) (x_i - x_j)}. \quad (3)$$

Note that for $x_i, x_j \in B$ and $\overrightarrow{x_i x_j} \not\subseteq B$, the distance between $x_i$ and $x_j$ is usually different from $D_M(B)(x_i, x_j)$. This is because some parts of $\overrightarrow{x_i x_j}$ may lie in influential regions so their lengths should be calculated via the related local metrics.

For any $x_i \in U$ and $x_j \in U$, its line segment $\overrightarrow{x_i x_j}$ may intersect with multiple influential regions and the background region. Therefore, we calculate the distance between $x_i$ and $x_j$ as the sum of lengths of line segments in each region. More precisely, as defined below, the distance is the sum of lengths of intersection of $\overrightarrow{x_i x_j}$ and influential regions, plus the length of intersection of $\overrightarrow{x_i x_j}$ and the background region.

**Definition 5.** The length of intersection of a line segment $\overrightarrow{x_i x_j}$ and an influential region $A_s$ is defined as $l(A_s \cap \overrightarrow{x_i x_j}; M(A_s))$, where $\cap$ denotes the intersection operator. The length of the intersection of a line segment $\overrightarrow{x_i x_j}$ and the background region $B$ is defined as

$$l(B \cap \overrightarrow{x_i x_j}; M(B)) = l(\overrightarrow{x_i x_j}; M(B)) - l(\bigcup_{s=1,\ldots,S} (A_s \cap \overrightarrow{x_i x_j}); M(B)),$$

where $\bigcup_{s=1,\ldots,S} (A_s \cap \overrightarrow{x_i x_j})$ denotes the union of intersections between the line segment and all influential regions.

**Definition 6.** The length of line segment $\overrightarrow{x_i x_j}$ is defined as

$$l(\overrightarrow{x_i x_j}; M(\overrightarrow{x_i x_j})) = \sqrt{(x_i - x_j)^T M(\overrightarrow{x_i x_j}) (x_i - x_j)}$$

$$= l(B \cap \overrightarrow{x_i x_j}; M(B)) + \sum_s l(A_s \cap \overrightarrow{x_i x_j}; M(A_s)),$$

1. Since influential regions are restricted to be ball-shaped and a ball is a convex set, $\overrightarrow{x_i x_j}$ would lie in the ball for any $x_i$ and $x_j$ inside the ball.
and the influential ball via the one-variable quadratic equation and the influential region are represented as Definition 7. Below.

If the line intersects with the ball (Fig. 3.3-3.8), we will calculate the tangent to the influential ball (Fig. 3.2), the length is zero. If Fig. 3. If the line length of intersection with influential regions, as illustrated in We start by providing an intuitive explanation of calculating the influential Regions 3.1 Calculation of the Length of Intersection with Influential Regions

Next, we figure out the relationship between the line \( x_i, x_j \) and the influential ball via the one-variable quadratic equation and calculate \( \lambda_u, \lambda_v \) when they exist. \( x_i, x_j \) intersects with \( A_s \) if we can find \( u, v \) that lie on the surface of the ball, which is equivalent to solving the following quadratic equation in one variable \( \lambda \):

\[
||x_i + \lambda(x_j - x_i) - o_s||^2 = r_s^2.
\]

If the discriminant of (6) is positive, then \( u, v \) exist and the solutions \( \lambda_{u,ij} \leq \lambda_{v,ij} \) are given by the quadratic equation

\[
\begin{align*}
\lambda_{u,ij}^s &= -\frac{2(x_j - x_i)^T (x_i - o_s) + \sqrt{\Delta}}{2(x_j - x_i)^T (x_j - x_i)}, \\
\lambda_{v,ij}^s &= -\frac{2(x_j - x_i)^T (x_i - o_s) - \sqrt{\Delta}}{2(x_j - x_i)^T (x_j - x_i)}, \\
\Delta &= [2(x_j - x_i)^T (x_i - o_s)]^2 \\
&- 4[(x_j - x_i)^T (x_j - x_i)][(x_i - o_s)^T (x_i - o_s) - r_s^2].
\end{align*}
\]

For simplicity, we will drop the superscript \( s \) and subscript \( ij \).

Last, we calculate \( \lambda_u, \lambda_v \) based on \( \lambda_u, \lambda_v \). Since \( 0 \leq \lambda_u \leq \lambda_v \leq 1 \), we set \( \lambda_u = \lambda_v \) if and only if \( \lambda_u \in [0, 1] \) and similarly for \( \lambda_v \). In other words, we set \( \lambda_u, \lambda_v \) as follows: \( \lambda_p = \min(\max(\lambda_u, 0), 1), \lambda_q = \min(\max(\lambda_v, 0), 1) \). Details are given in Table 1.

3.2 Calculation of the Length of Intersection Using Local Metrics

Proposition 2. In the case of non-overlapping influential regions, i.e. \( A_j \cap A_j = \emptyset, \forall i \neq j \),

\[
D_M(x_i, x_j) = \gamma_i \sqrt{[(x_i - x_j)^T M(B)(x_i - x_j)] + \sum_s \gamma_s \sqrt{[(x_i - x_j)^T M(A_s)(x_i - x_j)]},
\]

where \( \gamma_i \) is defined as the intersection ratio of the background region and \( \gamma_0 = 1 - \sum_s \gamma_s \).

Proof:

\[
\begin{align*}
D_M(x_i, x_j) &= l(x_i, x_j; M(B)) - l(\bigcup_{s=1}^{S} (A_s \cap x_i, x_j); M(B)) \\
&= \sum_s l(A_s \cap x_i, x_j; M(A_s)) \\
&= (1 - \sum_s \gamma_s) \sqrt{[(x_i - x_j)^T M(B)(x_i - x_j)] + \sum_s \gamma_s \sqrt{[(x_i - x_j)^T M(A_s)(x_i - x_j)]}. \tag{7}
\end{align*}
\]

Table 1

<table>
<thead>
<tr>
<th>Illustration</th>
<th>( \lambda_{u,ij} )</th>
<th>( \lambda_{v,ij} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 3.3</td>
<td>( \lambda_u &lt; 0 \Rightarrow \lambda_p = 0 )</td>
<td>( \lambda_v &lt; 0 \Rightarrow \lambda_q = 0 )</td>
</tr>
<tr>
<td>Fig. 3.4</td>
<td>( \lambda_u &gt; 1 \Rightarrow \lambda_p = 1 )</td>
<td>( \lambda_v &gt; 1 \Rightarrow \lambda_q = 1 )</td>
</tr>
<tr>
<td>Fig. 3.5</td>
<td>( \lambda_u &lt; 0 \Rightarrow \lambda_p = 0 )</td>
<td>( \lambda_v &gt; 1 \Rightarrow \lambda_q = 1 )</td>
</tr>
<tr>
<td>Fig. 3.6</td>
<td>( 0 \leq \lambda_u \leq 1 \Rightarrow \lambda_p = \lambda_u )</td>
<td>( 0 \leq \lambda_v \leq 1 \Rightarrow \lambda_q = \lambda_v )</td>
</tr>
<tr>
<td>Fig. 3.7</td>
<td>( 0 \leq \lambda_u \leq 1 \Rightarrow \lambda_p = \lambda_u )</td>
<td>( 0 \leq \lambda_v \leq 1 \Rightarrow \lambda_q = \lambda_v )</td>
</tr>
<tr>
<td>Fig. 3.8</td>
<td>( \lambda_u &lt; 0 \Rightarrow \lambda_p = 0 )</td>
<td>( 0 \leq \lambda_v \leq 1 \Rightarrow \lambda_q = \lambda_v )</td>
</tr>
</tbody>
</table>

Fig. 3. The positions of \( u, v \) (intersection points between line \( x_i, x_j \) and the influential region \( A_j \)) and \( p, q \) (intersection points between line segment \( x_i, x_j \) and \( A_j \)) under different situations. \( h \) is the middle point of line segment \( pq \).
of location parameters \((\alpha, r)\) leads to overlap, then we will skip this update and continue on learning other parameters.

4 Classifier and Learnability

In this paper, we select Lipschitz continuous functions as our classifiers since they are a family of smooth functions which are learnable [24]. Based on the resultant learning bounds, we obtain the regularization terms in order to improve the classifier’s generalization ability.

4.1 Classifier

To start with, we can see that the following classifier gives the same classification result as 1-NN:

\[
f(x) = \min \{D_{set}(x, X^-) - D_{set}(x, X^+),
\]

where \(D_{set}(x, X^-/+ = \{D(x, x_i) | \forall x_i \in \text{negative class / positive class} \}\) and \(D(x, x_j)\) denotes the distance between \(x_i\) and \(x_j\) defined by any metric. \(f(x) < 0\) indicates that \(x\) belongs to negative class and \(f(x) > 0\) indicates that \(x\) belongs to positive class.

In this paper, in order to achieve robustness to noisy instances and incorporate more flexible distance metrics, we extend the above equation by considering more nearby instances as follows:

\[
f(x) = \frac{1}{K} \sum_{k=1}^{K} D_{k}(x, X^-) - \frac{1}{K} \sum_{k=1}^{K} D_{k}(x, X^+), \tag{8}
\]

where \(D_{k}(x, X) = \{D(x, x_i) | \forall x_i \in X \}\) denotes the \(k\)th smallest element of the distance set \(\{D(x, x_i) | \forall x_i \in X \}\). This function will be used as the classifier in our algorithm.

For multiclass classification, the result will be given by

\[y = \arg\min_{c} \sum_{k=1}^{K} D_{k}(x, X^c),\]

where \(X^c\) denotes the training instances of class \(c\). It gives the same classification result as (8) in the binary case.

4.2 Learnability of the Classifier with Local Metrics

We will discuss learnability of functions based on the Lipschitz constant, which characterizes the smoothness of a function. The smaller the Lipschitz constant is, the smoother the function is.

Definition 8. [25] Let \((X, \rho_X), (Y, \rho_Y)\) be two metric spaces. The Lipschitz constant of a function \(f\) is

\[\text{Lip}(f) = \min \{C \in \mathbb{R} | \forall x_i, x_j \in X, x_i \neq x_j, \rho_Y(f(x_i), f(x_j)) \leq C \rho_X(x_i, x_j)\}\]

\[= \max_{x_i, x_j \in X, x_i \neq x_j} \frac{\rho_Y(f(x_i), f(x_j))}{\rho_X(x_i, x_j)}\]

Proposition 3. [25] Let \(\text{Lip}(f) \leq L_f\) and \(\text{Lip}(g) \leq L_g\), then

(a) \(\text{Lip}(f + g) \leq L_f + L_g\);
(b) \(\text{Lip}(fg) \leq L_f + L_g\);
(c) \(\text{Lip}(af) \leq |a| L_f\), where \(a\) is a constant.

Proposition 4. Let \(\text{Lip}(f_k(x)) \leq L, k = 1, \ldots, N\), then, for any \(K \leq N\), \(\text{Lip}(\sum_{k=1}^{K} f_k(x))\) is bounded by \(K \max_k L\), where \(f_k(x)\) denotes the \(k\)th smallest element of the set \(\{f_k(x), k = 1, \ldots, K\}\).

Proof. \(\forall x_i, x_j \in X, k \in \{1, \ldots, N\}\)

\[
\sum_{k=1}^{K} f_k(x_i) = \sum_{k=1}^{K} \{f_k(x_j) + f_k(x_j + (x_i - x_j)) - f_k(x_j)\}\]

\[
\leq \sum_{k=1}^{K} f_k(x_j) + L ||x_j + (x_i - x_j) - x_j||\]

\[\leq \sum_{k=1}^{K} f_k(x_j) + L ||x_i - x_j||\]

\[= \sum_{k=1}^{K} f_k(x_j) + KL ||x_i - x_j||\]

Based on the definition of Lipschitz constant, the proposition is proved.

Lemma 1. With distance defined in (7), the Lipschitz constant of the classifier specified in (8) is bounded by \(L = 2(\sum_s \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F})\), where \(\cdot \|_F\) denotes the matrix Frobenius norm.

Proof. Let \(d_M(x, x_k)\) denote the Mahalanobis distance with metric \(M\), i.e.

\[d_M(x, x_k) = \sqrt{(x - x_k)^T M (x - x_k)},\]

and \(d_I(x, x_k)\) denotes the Euclidean distance with the identity matrix \(I\).

The Mahalanobis distance \(d_M(x, x_k)\) has the Lipschitz constant of \(\sqrt{\|M\|_F}\) as follows:

\[\text{Lip}(d_M(x, x_k)) = \sup_{x_a, x_b \in X, x_a \neq x_b} \frac{d_M(x_a, x_b) - d_M(x_a, x_b)}{d_I(x_a, x_b)}\]

\[\leq \sup_{x_a, x_b \in X, x_a \neq x_b} \frac{d_M(x_a, x_b)}{d_I(x_a, x_b)}\]

\[\leq \sup_{x_a, x_b \in X, x_a \neq x_b} \frac{d_I(x_a, x_b) \sqrt{\|M\|_F}}{d_I(x_a, x_b)}\]

\[= \sqrt{\|M\|_F},\]

where the first inequality follows the triangle inequality of distance, and the second inequality is based on the Cauchy-Schwarz inequality and the fact that Frobenius norm is compatible with the vector l2 norm.

According to the definition of distance in (7), we have

\[D_M(x, x_k) = \sum_{s} \gamma_s d_M(A_s)(x, x_k) + \gamma_b d_M(B)(x, x_k)\]

as \(\gamma_s, \gamma_b \leq 1\). From Proposition 3, we get that

\[\text{Lip}(D_M(x, x_k)) \leq \sum_{s} \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F} + \sqrt{\|M(B)\|_F} + \sqrt{\|M(B)\|_F}.

Based on the Lipschitz constant of \(D_M(x, x_k)\) and the composition property illustrated in Proposition 4,

\[\text{Lip}(\sum_{k=1}^{K} (D_M(x, x_k), k = 1, \ldots, K)} \leq K \left\{ \sum_{s} \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F} \right\}.

\]
Finally, based on Proposition 3, $f(x)$ in (8) is bounded by
\[ 2\left(\sum_{k} \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F}\right). \]

Combining Lemma 1 and Corollary 2 of [24], we can obtain the following Corollary.

**Corollary 1.** Let metric space $X$ have doubling dimension $d$ and let $F$ be the collection of real valued functions over $X$ with the Lipschitz constant at most $L$. Then for any $f \in F$ that classify correctly on all but $k$ examples, we have with probability at least $1 - \delta$
\[
P(\{x, t\} : \text{sign}(f(x)) \neq t) \leq \frac{k}{n} + \sqrt{\frac{2}{n} \left( e \ln(34en/c) \log(578n + 4\ln(4/d)) \right)},
\]
where $n$ denotes the sample size, $t \in \{-1, 1\}$ denotes the label, and $c \leq (16L \text{diam}(X))^d \text{diam}(X)$
\[
= \left(32\sum_{s} \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F}\right) \text{diam}(X) \text{diam}(X).
\]
diam denotes the diameter of the space and $\text{diam}(X)$ denotes doubling dimension; precise definitions can be found in [24].

The above learning bound illustrates that the generalization ability, i.e. the difference between the expected error $P(\{x, t\} : \text{sign}(f(x)) \neq t)$ and the empirical error $k/n$, can be improved by reducing the value of $\sum_{s} \sqrt{\|M(A_s)\|_F} + \sqrt{\|M(B)\|_F}$. Since the square function is monotonically increasing, we would instead reduce $\sum_{s} \|M(A_s)\|_F + \|M(B)\|_F$. In other words, $\sum_{s} \|M(A_s)\|_F + \|M(B)\|_F$ would be used as the regularization term to improve the generalization ability of the classifier.

## 5 Optimization Problem

### 5.1 Objective Function

In order to obtain low training error and good generalization ability, we propose the following optimization problem, where the objective function consists of a sum of hinge loss and the regularization term $\sum_{s} \|M(A_s)\|_F + \|M(B)\|_F$:
\[
\min_{\Theta, M(A_s), M(B), \alpha, r} \frac{1}{N_i} \sum_{(i, j)} \xi_{ij} + \frac{1}{N_2} \sum_{(m, n)} \xi_{mn} + \alpha \sum_{s} \|M(A_s)\|_F + \|M(B)\|_F
\]
s.t. $D_M(x_i, x_j) \leq 1 - C + \xi_{ij}, D_M(x_i, x_n) \geq 1 + C + \xi_{mn}$
\[
\xi_{ij}, \xi_{mn} \geq 0, M \in M_s,
\]
i, $m = 1, \ldots, N, j \rightarrow i, n \rightarrow m,
\]
where $\Theta = \{M(A_s), M(B), \alpha, r\}$ denotes the set of parameters to be optimized; $j \rightarrow i$ denotes that $x_i$ is $x_j$'s $K$ nearest neighbor comparing against all instances in the same class; $m \rightarrow n$ denotes that $x_n$ is $x_m$'s $K$ nearest neighbor comparing against all instances in the different class; $C$ is a constant which has the intuition of margin; $\xi_{ij}$ and $\xi_{mn}$ denote the error caused by margin violation; $N_i, N_2$ denote the number of training samples, pairs $(i, j)$, and pairs $(m, n)$ respectively; $\alpha$ is a trade-off parameter between the margin loss and the regularization terms. This optimization formula is suitable for both binary and multi-class tasks. In the proposed algorithm, we will learn the locations of influential regions $(\alpha_s, r_s)$ and the metrics of influential/background regions $(M(B), M(A_s))$ under the same framework.

### 5.2 Gradient Descent

With $D_{M(A_s)}$ and $D_{M(B)}$ being the Mahalanobis distances, the optimization problem is convex even when $\alpha, r$ are fixed and only $M(A_s)$ and $M(B)$ are updated. Therefore, we adopt the gradient descent algorithm:
\[
\Theta^{t+1} = \Theta^t - \beta \frac{\partial g}{\partial \Theta}_{|\Theta^{t}},
\]
where $\beta$ is the learning rate, and the superscript $t$ denotes the time step during optimization.

The objective function $g$ is
\[
g = \frac{1}{N_1} \left[ D_M(x_i, x_j) - (1 - C) \right]_{+} + \alpha \sum_{s} \|M(A_s)\|_F
\]
\[
+ \frac{1}{N_2} \left[ 1 + C - D_M(x_i, x_n) \right]_{+} + \alpha \|M(B)\|_F,
\]
and $\gamma_{\Theta}(\alpha_s, r_s) = 1 - \sum_{s} \gamma_{\Theta}(\alpha_s, r_s)$. Here, $\gamma_{\Theta}$ is written as $\gamma_{\Theta}(\alpha_s, r_s)$ to remind us that $\gamma_{\Theta}$ is a function of the location parameters $\alpha$ and $r$.

The gradient of $g$ with respect to parameters $\alpha, r$ is
\[
\frac{\partial g}{\partial \Theta}_{|\Theta^{t}} = \frac{1}{N_1} \sum_{(i, j)} 1[D_M(x_i, x_j) - (1 - C) > 0] \frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}},
\]
\[
- \frac{1}{N_2} \sum_{(m, n)} 1[1 + C - D_M(x_m, x_n) > 0] \frac{\partial D_M(x_m, x_n)}{\partial \Theta}_{|\Theta^{t}}.
\]

If the gradient is with respect to $M(B)$ and $M(A_s)$, then the shrinkage term of $\frac{\partial M(B)}{\partial \Theta}_{|\Theta^{t}}$ or $\frac{\partial M(A_s)}{\partial \Theta}_{|\Theta^{t}}$ should be added into the above formula.

Now we will calculate $\frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}}$ for the parameters $M(A_s), M(B), \alpha_s, r_s$ separately:
\[
\frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}} = \frac{\gamma_{\Theta}(\alpha_s, r_s)}{2} \times
\]
\[
(1 - (x_i - x_j)^T M(A_s)(x_i - x_j)^{-1/2} (x_i - x_j)(x_i - x_j)^T)^T;
\]
\[
\frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}} = \frac{1}{2} \frac{\gamma_{\Theta}(\alpha_s, r_s)}{0} > 0 \gamma_{\Theta}(\alpha_s, r_s) \times
\]
\[
(1 - (x_i - x_j)^T M(B)(x_i - x_j)^{-1/2} (x_i - x_j)(x_i - x_j)^T)^T;
\]
\[
\frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}} = \frac{1}{2} \frac{\gamma_{\Theta}(\alpha_s, r_s)}{0} > 0 \gamma_{\Theta}(\alpha_s, r_s) \times
\]
\[
(1 - (x_i - x_j)^T M(A_s)(x_i - x_j)^{-1/2} (x_i - x_j)(x_i - x_j)^T)^T;
\]
where $\frac{\partial \Theta}{\partial \Theta}$ could be obtained as illustrated in Table 2;
\[
\frac{\partial D_M(x_i, x_j)}{\partial \Theta}_{|\Theta^{t}} = \frac{1}{2} \frac{\gamma_{\Theta}(\alpha_s, r_s)}{0} > 0 \gamma_{\Theta}(\alpha_s, r_s) \times
\]
\[
(1 - (x_i - x_j)^T M(B)(x_i - x_j)^{-1/2} (x_i - x_j)(x_i - x_j)^T)^T;
\]
where $\frac{\partial \Theta}{\partial \Theta}$ could be obtained as illustrated in Table 2.

Initial values are crucial for non-convex optimization problems. We adopt a heuristic method to initialize the parameters as
follows. 1) Extract local discriminative direction \( h(x) \in \mathbb{R}^F \) for each training instance \( x_i \), where \( F \) indicates the number of features of \( x_i \):

\[
h(x_i)[f] = \sum_{k=1}^{\lambda_u} [x_k[f] - x_i[f]] - \sum_{j=1}^{\lambda_u} [x_j[f] - x_i[f]],
\]

where \( x[f] \) indicates the \( f \)-th dimension of vector \( x \). 2) Perform nonparametric clustering: The Dirichlet process Gaussian mixture model is applied to the augmented feature vector \( [x, h(x)] \) to group instances into clusters; the number of clusters, and hence the number of influential regions, is automatically decided by the clustering algorithm. 3) Initialize the parameters: Cluster centers are initialized as \( \alpha_{si} \); the 80th percentile of the distance between samples and the cluster center is set as initial value of \( r_s \); the local metric is set as \( M(A_s) = I + 0.1 \times \text{diag}([\text{mean}(h(x), x \in \text{cluster } s)]) \), where \( \text{diag} \) is an operator which returns a square diagonal matrix with elements of the input vector on the main diagonal. The initialization process is carried out from the largest cluster to the smallest one. If a later influential region overlaps with an earlier one, the later region will be shrunk, or even deleted, until no overlap exists.

### 6 Experiments

#### 6.1 Toy Example

To visualize the learned parameters, we consider a toy data set for binary classification consisting of 80 instances generated from a two-component Gaussian mixture model. 40 instances in the positive and 40 instances in the negative class are sampled from \( \frac{1}{2}N([-1, 0], \frac{1}{2}I) + \frac{1}{2}N([1, 0], \frac{1}{2}I) \) and \( \frac{1}{2}N([-1, 2], \frac{1}{2}I) + \frac{1}{2}N([3, 0], \frac{1}{2}I) \) respectively. Parameters in our algorithm are set as follows: \( \alpha \) and \( C \) in the optimization formula are 0.1 and 0.5 respectively; the number of clusters used for initializing the parameters is 2; the gradient descent algorithm stops after 50 iterations. For illustration purpose, overlap detection has not been conducted on the toy example.

In Figs. 4a-4c, we learn one parameter from \( \{M(A), \alpha, r\} \) at each time, fixing the other parameters. Take Fig. 4a (left) as an example. Since \( M(A_1) = M(A_2) = 2I \) and \( M(B) = I \), the influential regions act as enlarging the local distance. In this case, we see that the centers of \( A1 \) and \( A2 \) move to the inter-class region. This phenomenon could be explained as follows. For a line segment that lies in an inter-class region and violates the margin constraint, i.e. \( D_M(x_m, x_n) < 1 + C \), the direction of gradient descent is same as that of \( \partial D_M(x_m, x_n) / \partial \theta \). As \( D_M(A_1)(x_m, x_n) > D_M(B)(x_m, x_n) \), \( \partial D_M(x_m, x_n) / \partial \theta \) has the same direction as \( \partial \gamma / \partial \theta \), which, according to Table 2, is the direction of \( \partial \gamma / \partial \theta \), \( \partial o_h / \partial \theta \), \( \partial o_O / \partial \theta \) in Fig. 3 depending on the value of \( \gamma \). In other words, the margin-violated inter-class line segments will pull the influential regions towards the inter-class region. At the same time, for an intra-class line segment that violates the margin constraint, i.e. \( D_M(x_m, x_j) > 1 - C \), the direction of gradient descent is opposite to that of \( \partial D_M(x_m, x_n) / \partial \theta \), and hence opposite to \( \partial \gamma / \partial \theta \), \( \partial o_h / \partial \theta \), \( \partial o_O / \partial \theta \). That is, the margin-violated intra-class line segments will push the influential regions away from the intra-class region. In summary, as illustrated in Fig. 4a (left), when the influential regions have the effect of ‘enlarging’ distance, \( o \) move to the inter-class region. Similar reasoning applies to Fig. 4a (right), 4b, and 4c. In Fig. 4d, \( M(A), \alpha, r \) are learned simultaneously. As expected, the influential regions focus on inter-class samples by moving towards the inter-class region, increasing the region size, and enlarging the local distance in the direction that is nearly perpendicular to the decision boundary.

The toy example demonstrates that the gradient learning has a clear geometric interpretation.

#### 6.2 Real Data

We compare our algorithm with twelve established metric learning algorithms from three categories: (1) the most cited algorithms, including large margin nearest neighbor (LMNN) [2] and information theoretic metric learning (ITML) [3]; (2) local metric learning algorithms, including multiple-metric large margin nearest neighbor (mMLNN) [2], parametric local metric learning (PLML) [17], reduced-rank local distance metric learning (R2LML) [18], and local discriminative distance metrics ensemble learning (LDDM) [26]; (3) the state-of-the-art metric learning algorithms, including distance metric learning with eigenvalue optimization (DMLE) [27], sparse compositional metric learning (SCML) [20], stochastic neighbor compression (SNC) [28], regressive virtual metric learning (RVML) [29], geometric mean metric learning (GMMML) [30], and supervised distance metric learning through maximization of the Jeffrey divergence (DMLMJ) [31]. LMNN and ITML are implemented using the metric-learn toolbox\(^2\); mMLNN, PLML, R2LML, LDDM, DMLE, SCML, SNC, RVML, GMMML and DMLMJ are implemented using the authors’ code.

We conduct binary classification on 14 data sets and multiple-classification on 6 data sets, all of which are publicly available from UCI\(^3\) and LibSVM\(^4\). For binary classification, we use data sets Australian, Breastcancer, Diabetes, Fourclass,

\(^2\) 2. https://all-umass.github.io/metric-learn/

---

**Table 2**

<table>
<thead>
<tr>
<th>( \lambda_u, \lambda_o )</th>
<th>( \gamma )</th>
<th>partial gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_u &lt; 0, \lambda_o &lt; 0 )</td>
<td>0</td>
<td>( \partial \gamma / \partial \theta = 0 )</td>
</tr>
<tr>
<td>( \lambda_u &lt; 0, \lambda_o &gt; 1 )</td>
<td>1</td>
<td>( \partial \gamma / \partial \theta = 0 )</td>
</tr>
<tr>
<td>( \lambda_u &gt; 1, \lambda_o &gt; 1 )</td>
<td>0</td>
<td>( \partial \gamma / \partial \theta = 0 )</td>
</tr>
<tr>
<td>( 0 \leq \lambda_u \leq 1, 0 \leq \lambda_o \leq 1 )</td>
<td>( \gamma - \lambda_u )</td>
<td>( \frac{\partial \gamma}{\partial \theta} = \frac{2 \gamma - \frac{1}{2} - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o)}{r_s} - 4 \Delta - \frac{1}{2} \left( \frac{1}{r_s} (x_m - x_o)(x_n - x_o) \right) - 4 \Delta - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o) )</td>
</tr>
<tr>
<td>( \lambda_u &lt; 0, 0 \leq \lambda_o \leq 1 )</td>
<td>( \lambda_u )</td>
<td>( \frac{\partial \gamma}{\partial \theta} = \frac{2 \gamma - \frac{1}{2} - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o)}{r_s} - 4 \Delta - \frac{1}{2} \left( \frac{1}{r_s} (x_m - x_o)(x_n - x_o) \right) - 4 \Delta - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o) )</td>
</tr>
<tr>
<td>( 0 \leq \lambda_u \leq 1, \lambda_o &gt; 1 )</td>
<td>( 1 - \lambda_u )</td>
<td>( \frac{\partial \gamma}{\partial \theta} = \frac{2 \gamma - \frac{1}{2} - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o)}{r_s} - 4 \Delta - \frac{1}{2} \left( \frac{1}{r_s} (x_m - x_o)(x_n - x_o) \right) - 4 \Delta - \frac{1}{2} \frac{1}{r_s} (x_m - x_o)(x_n - x_o) )</td>
</tr>
</tbody>
</table>
Germanumber, Haberman, Heart, ILPD, Liverdisorders, Monk1, Pima, Planning, Vote, and WDBC; for multiple-class, we use Cleveland, Glass, Iris, Newyorkheart, Tae, and Winequality (red). All data sets are pre-processed by firstly subtracting the mean and dividing by the standard deviation, and then normalizing the L2-norm of each instance to one.

For each data set, 60% instances are randomly selected as training samples and the rest for testing. This process is repeated 10 times and the mean accuracy and the standard deviation are reported. We use 10-fold cross-validation to select the trade-off parameters in the compared algorithms, namely the regularization parameter of LMNN (from \{0.1, 0.3, 0.5, 0.7, 0.9\}, \(\gamma\) in ITML (from \{0.25, 0.5, 1, 2, 4\}, \(t\) in GMML (from \{0.1, 0.3, 0.5, 0.7, 0.9\}) and \(\lambda\) in RVML (from \{10^{-2}, 10^{-4}, \ldots, 1\}). All other parameters are set as default.

For our algorithm, we set the parameters as follows: \(\alpha\) and \(C\) in the optimization formula are 0.1 and 0.5 respectively; \(K\) in the classifier is 10. The number of influential regions in our algorithm is determined via Dirichlet process Gaussian mixture model and it is implemented with PRML toolbox 3.

Results for binary classification and multiclass classification are shown in Table 3. The proposed algorithm achieves the highest average accuracy on both tasks. Out of 20 data sets, LMLIR outperforms all other methods on ten data sets out and none of the other algorithms performs the best in more than two data sets. In cases where our algorithm is not leading, the difference to the optimal method is relatively small. Such encouraging results demonstrate the effectiveness of our proposed method.

### References


Fig. 4. Illustration of parameter learning using a toy data set. This figure is best viewed in color.