CHAPTER 5
METRIC LEARNING

5.1 Introduction

Metric learning aims to learn an appropriate distance/similarity function for a given problem. Because many learning problems involve some notion of distance or similarity, metric learning can be applied in a variety of settings and has become a popular problem for many learning tasks.

Metric learning plays a crucial role in machine learning research, as many machine learning algorithms are based on some distance metrics. Metric learning for classification tasks has a long history that can be dated back to some early, work for nearest neighbor classifiers more than two decades ago. More recent work includes learning locally adaptive metrics [Domeniconi C, 2002], [Friedman, J, 1994], [Hastie T, 1996] and global Mahalanobis metrics [Goldberger J, 2005].

Metric learning methods have also been proposed for semi-supervised clustering tasks, with supervisory information available in the form of either limited labeled data or pair wise constraints [Bar-Hillel A, 2003], [Chang H, 2004]. Instead of devising metric learning methods for specific classification or clustering algorithms, it is also possible to devise generic metric learning methods that can be used for both classification and clustering tasks.

In this thesis, one wants to devise a more flexible metric learning method that can perform locally linear transformations, yet it is as efficient as algorithms for learning globally linear metrics. The method is referred to as locally smooth metric learning (LSML), which is inspired by the method of moving least squares for function approximation from point wise scattered data [Levin D, 1998]. However, the optimization problem for LSML is significantly different from that for previous methods, because one can formulate the problem under a
regularization framework with which unlabeled data can also play a role. This is particularly crucial when labeled data are scarce.

5.2. The Method

5.2.1. Basic concepts of Locally Smooth Metrics

Let $X = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_n\}$ denote $n$ data points in a $d$-dimensional input space $R^d$. For semi-supervised learning, one can assume that supervisory side information is available for the first $l$ data points. The side information, represented by a set $S$, can be in the form of labels for data points or pairwise constraints between data points. Intuitively, it is wanted to perform local transformations on the data points such that the points that belong to the same class or are considered similar according to the similarity constraints will get closer after they are transformed. Similar to [Chang H, 2004], one can resort to locally linear transformations.

$$f_i(x) = A_i x + b_i, \ldots \quad (1)$$

where $A_i$ denotes a $d \times d$ transformation matrix and $b_i$ a $d$-dimensional translation vector. After metric learning, $x$ is transformed to $f_i(x)$. Note that a different locally linear transformation $f_i(\cdot; A_i, b_i)$ is associated with each input data point $x_i$. It would be desirable if $f_i(\cdot; A_i, b_i)$ changes smoothly in the input space so that the transformed data points after metric learning can preserve the intra-class topological structure of the original data. Since there are $n$ locally linear transformations, one for each input data point, one can have a large number of parameters to determine.

It is proceeded to devise the metric learning algorithm more formally. One can generate the transitive and reflective closure $S$ from initial value $S_0$. For each point pair $(x_r, x_s) \in S$, one can apply a linear transformation to the vector $(x_s - x_r)$ to give $A_r (x_s - x_r) + c_r$ for some $d \times d$ matrix $A_r$ and $d$-dimensional vector $c_r$. If a data point is involved in more than one point pair, one can consider the transformation for each pair separately. The same linear transformation is also applied to every data point $x_i$ in the neighborhood set $N_r$ of $x_r$. In other words, every data point $x_i \in N_r$ is transformed to
\[ Y_i = A_r (x_i - x_r) + c_r + x_r \]
\[ = x_i + (A_r - I) x_i + b_r, \]

where \( b_r = (I - A_r)x_r + c_r \) is the translation vector for all points \( x_i \)'s in \( N_r \).

However, a data point \( x_i \) may belong to multiple neighborhood sets corresponding to different point pairs in \( S \). Thus, the new location \( y_i \) of \( x_i \) is the overall transformation effected by possibly all similar point pairs (and hence neighborhood sets):

\[ y_i = x_i + \sum_{(x_j, x_s) \in S} \pi_{ri} [(A_r - I) x_j + b_r] \]

where \( \pi_{ri} = 1 \) if \( x_i \in N_r \) and 0 otherwise.

Let \( m \) denote the number of point pairs in \( S \). Thus a total of \( m \) different transformations have to be estimated from the point pairs in \( S \), requiring \( O(md^2) \) transformation parameters in \( \{A_r\} \) and \( \{b_r\} \). When \( m \) is small compared with the dimensionality \( d \), one cannot estimate the \( O(md^2) \) transformation parameters accurately. One way to get around this problem is to focus on a more restrictive set of linear transformations. The simplest case is to allow only translation, which can be described by \( md \) parameters. Obviously, translating all data points in a neighborhood set by the same amount leads to no change in the inter-point distances. Although some data points may fall into multiple neighborhood sets and hence this phenomenon does not hold, one wants to incorporate an extra degree of freedom by changing the neighborhood sets to Gaussian neighborhood functions. More specifically, one can set \( A_r \) to the identity matrix \( I \) and express the new location \( y_i \) of \( x_i \) as

\[ y_i = x_i + \sum_{(x_j, x_s) \in S} \pi_{ri} b_r \tag{2} \]

where \( \pi_{ri} \) is a Gaussian function defined as

\[ \pi_{ri} = \exp \left[ -\frac{1}{2} (x_i - x_r)^T \sum_r^{-1} (x_i - x_r) \right] \]
with \( \sum_r \) being the covariance matrix. For simplicity, one can use a hyper-spherical Gaussian function, meaning that the covariance matrix is diagonal with all diagonal entries being \( \omega^2 \). Thus \( \pi_{ri} \) can be rewritten as \( \pi_{ri} = \exp\left(-\left\| x_i - x \right\|^2 / 2 \left( \omega^2 \right) \right) \). Note that (2) can be expressed as

\[
y_i = x_i + B\pi_i
\] ...

(3)

where \( B = [b_1, b_2, \ldots, b_m] \) is a \( dxm \) matrix and \( \pi_i = (\pi_{i1}, \pi_{i2}, \ldots, \pi_{im})^T \) is an \( m \)-dimensional column vector. For data points that are far away from all \( n \) points involved in \( S \) (and hence the centers of the neighborhoods), all \( \pi_{ri} \)'s are close to 0 and hence those points essentially do not move (since \( y_i \sim x_i \)).

One can formulate the optimization problem for finding the transformation parameters. The optimization criterion is defined as

\[
J = d_S + \lambda P
\] ...

(4)

where \( d_S \) is the sum of squared Euclidean distances for all similar pairs in the transformed space

\[
d_S = \sum_{(y_i, y_j) \in S} \left\| y_i - y_j \right\|^2
\]

and \( P \), a penalty term used to constrain the degree of transformation, is defined as

\[
P = \sum_i \sum_j N_\sigma(d_{ij})(q_{ij} - d_{ij})^2
\] ...

(5)

where \( q_{ij} = \left\| y_i - y_j \right\| \) \( yjk \) and \( d_{ij} = \left\| x_i - x_j \right\| \) represent the inter-point Euclidean distances in the transformed and original spaces, respectively. \( N_\sigma(d_{ij}) \) is again in the form of a Gaussian function, as \( N_\sigma(d_{ij}) = \exp \left(-d_{ij}^2 / \sigma^2 \right) \) with parameter \( \sigma \) specifying the spread of the Gaussian window. The regularization parameter \( \lambda > 0 \) in (4) determines the relative significance of the penalty term in the objective function for the optimization problem. Note that the optimization criterion in (4) is analogous to objective functions commonly used in energy minimization models such as deformable models, with the penalty term \( P \) playing the role of an internal energy term.
5.2.2. Regularized Moving Least Squares

The Moving Least Squares algorithm is a deformation technique that allows to compute a map \( f: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) from the transformation of a set of \( N \) pivot points \( p \) in the new positions \( q \). The map \( f \) is smooth (\( f \in C^2 \)), preserves the identity (for \( q=p \) the map is the identity) and ensures that the points \( p \) are transformed in the points \( q \) (\( f(p)=q \)). The work describes a simple optimization technique that allows to get, for each point of the \( \mathbb{R}^2 \) plane, the best linear transformation (affinity, similarity or rigid transformation) that moves the points according to a set of weights of the transformed \( p_i-q_i \).

To compute the locally linear transformations for all input data points, one formulates metric learning as a set of \( n \) optimization problems. One can extend the method of moving least squares [Levin D, 1998] to regularized moving least squares under a regularization framework. For each input data point \( x_i \), one computes its local affine transformation by minimizing the following objective function:

\[
J(A_i, b_i) = \sum_{j=1}^{l} \theta_{ij} \| f_i(x_j) - y_j \|^2 + \lambda \sum_{j=1}^{n} \omega_{ij} \| f_i(x_j) - f_i(x_j) \|^2 \quad \ldots \quad (6)
\]

The first term in the objective function is the moving least squares term. One can define the moving weight function that depends on both \( x_i \) and \( x_j \) as \( \Theta_{ij} = 1/\| x_i - x_j \|^2 \). For the first \( l \) data points, the target location of \( x_j \) after metric learning is denoted as \( y_j \) which can be computed from the input data and the side information. Although it has been demonstrated that the method of moving least squares works very well for interpolation and the learned functions \( f_i(\cdot ; A_i, b_i) \) vary smoothly [Levin D, 1998], one can notice that performance degrades when the data points with side information are not evenly distributed in the input space. The second term in Equation (2) is to address this problem through regularization by incorporating unlabeled data. It restricts the degree of local transformation to preserve local neighborhood relationships. Like graph-based semi-supervised learning methods, one can consider the \( n \) points as corresponding to a graph consisting of \( n \) nodes, with a weight between every two nodes. Since the weights \( wij \) are similar to the edge weights in graph-based methods, one also defines \( wij \) as \( wij = \exp(-\| x_i - x_j \|^2) \).
x_j \| z/\sigma^2 \) with \( \sigma > 0 \) specifying the spread. To combine the two terms, the parameter \( \lambda > 0 \) controls the relative contribution of the regularization term in the objective function.

The target points \( y_j \) (\( j = 1 \ldots l \)) are set differently for different learning tasks. For classification tasks with partial label information, one possibility is to set \( y_j \) to the mean of all \( x_k \)'s with the same label as \( x_j \). For clustering tasks with some given pair wise similarity constraints, one may set \( y_j \) by pulling similar points towards each other.

5.2.3. Optimizing the Objective Functions

It can be noted that the objective function \( J(A_i, b_i) \) for the \( i \)th local transformation involves the parameters of all \( n \) local transformations, \( f_j(\cdot; A_j, b_j) \) (\( j = 1, \ldots, n \)). Since \( J(A_i, b_i) \) is quadratic in \( A_i \) and \( b_i \), in principle it is possible to obtain a closed-form solution for the parameters of all \( n \) transformations by solving a set of \( n \) equations. This approach is undesirable, though, as it requires inverting a possibly large \( n \times n \) matrix. One can propose here a more efficient alternative approach for obtaining an approximate solution.

More specifically, one can estimate \( f_i(\cdot; A_i, b_i) \) by minimizing a modified form of Equation (2), with the regularization term replaced by one that incorporates only the \( i - 1 \) local transformations already estimated in the previous steps. It can be first estimated the local transformations for the first \( l \) data points with side information available. For the remaining \( n - l \) local transformations, one can order them such that the transformations for the data points closer to the first \( l \) points are estimated before those that are farther away. This may be seen as a process of propagating the changes from the labeled points to the unlabeled points.

Let \( \hat{y}_j = \hat{f}_j(x_j) = \hat{A}_j x_j + \hat{b}_j \) denote the new location of \( x_j \) after transformation. Substituting Equation (1) into the modified form of Equation (2), one can have the following approximate objective function (6):

One can obtain a closed-form solution for \( \hat{A}_i \) and \( \hat{b}_i \) as:

\[
\hat{A}_i = (D_i - \bar{Y}_i X T_i)(C_i - \bar{x}_i X T_i) \quad \text{......... (7)}
\]
For each local affine transformation, the main computation is to find the pseudo inverse of a $d \times d$ matrix. Thus the algorithm is very efficient.

5.2.4. Modified LSML Algorithm and an Example

The Modified LSML algorithm is summarized as follows:

Input: data set $X = \{x_1, \ldots, x_n\}$, side information $S$;
Compute target points: $y_i$ ($i = 1, \ldots, l$);
Metric learning: sort the data points in $\{x_{l+1}, \ldots, x_n\}$;
For $i = 1$ to $n$ do
    Compute local affine transformation using (7), (8);
End
Output: $\hat{A}_i, \hat{b}_i$ ($i = 1, \ldots, n$).

There are only two free parameters in the modified LSML algorithm, $\lambda$ and $\sigma$. It is set $\sigma^2$ to be the average squared Euclidean distance between nearest neighbors in the input space and $\lambda$ to some value in $[0, 1]$ which can be determined by cross validation.

Let us first look at an illustrative example using our proposed LSML method on the well-known 2-moon data set, as shown in Figure 5.1(a). The solid black squares represent the labeled data points. One sets the goal of metric learning as moving the points with the same label towards their center (as indicated by the arrows) while preserving the moon structure. One first computes the locations of the target points, as shown with black squares in Figure 5.1(b) and (c). For illustration purpose, one sets the target locations to be some points along the way to the class centers. In Figure 5.1(b), the data set after metric learning has shorter moon arms. If the target
points are closer to the class centers, the data set after metric learning forms more compact classes, as shown in Figure 5.1(c), which are desirable for classification and clustering tasks. From the color code, one can see that the intra-class topological structure is well preserved after metric learning, showing that the learned functions \( f_i(\cdot; A_i, b_i)(i = 1, \ldots, n) \) vary smoothly along the arms of the moons. Figure 5.1(d) and (e) demonstrate the effect of the regularization term when only a single labeled point is available for each class. The data set after metric learning with \( \lambda = 0 \) (without regularization) is shown in Figure 5.1(e) and that with \( \lambda = 0.5 \) in Figure 5.1(f). As one can see, the regularization term plays an important role in preserving the structure of the data especially when the side information available is scarce and unevenly distributed.

![Figure 5.1: (a) and (d) original 2-moon data set.; (b), (c), (e) and (f) transformed data after metric learning.](image)
5.3 Experiments on Semi-Supervised Learning Tasks

In this section, one can illustrate through experiments how the proposed LSML method can be used for different semi supervised learning tasks, including classification and clustering. The quality of metric learning is assessed indirectly via the performance of the respective learning tasks.

5.3.1 Semi-Supervised Nearest Neighbor Classification

One first performs some experiments on the Iris plant data set from the UCI Machine Learning Repository. There are 150 data points from three classes with 50 points per class. The dimensionality of the input space is 4. For visualization, one can plot the data points in a 2-D space before and after metric learning based on the two leading principal components of the data, as shown in Figure 5.3. Figure 5.3(a) shows the original data set. Data points with the same point style and color belong to the same class. One class is well separated from the other two classes which are very close to each other. Figure 5.3(b) shows the metric learning result obtained by LMNN. One can see that there is less overlap between the two nearby classes. This is expected to lead to better classification results. Using LSML, Figure 5.3(c) shows even higher separability between the two nearby classes. The data points with the same class label have been transformed to the same location. Using 20 randomly generated training sets with each set having five labeled points per class, the average classification rates for a 3-nearest neighbor classifier are 91.52% for LMNN and 94.63% for LSML. With the training sets increased in size to 10 points per class, the average classification rates for LMNN and LSML increase to 95.83% and 95.08%, respectively.
Table 5.1: Statistical Data for Minist and Isolet Databases

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Minist</th>
<th>Isolet</th>
</tr>
</thead>
<tbody>
<tr>
<td># inputs</td>
<td>70000</td>
<td>7797</td>
</tr>
<tr>
<td># features</td>
<td>784</td>
<td>617</td>
</tr>
<tr>
<td># reduced dimensions</td>
<td>164</td>
<td>172</td>
</tr>
<tr>
<td># training examples</td>
<td>60000</td>
<td>6238</td>
</tr>
<tr>
<td># testing examples</td>
<td>10000</td>
<td>1559</td>
</tr>
</tbody>
</table>

This shows that modified LSML is particularly good when labeled data are scarce. One can further perform more classification experiments on two larger data sets. One data set contains handwritten digits from the MNIST database. The digits in the database have been size-normalized and centered to $28 \times 28$ gray level images. In our experiments, one can randomly choose 2,000 images for digits ‘0’–‘4’ from a total of 60,000 digit images in the MNIST training set. Another data set is the isolet data set from the UCI Machine Learning Repository, which contains 7,797 isolated spoken English letters belonging to 26 classes with each letter
represented as a 617-dimensional vector. For both data sets, since the original features are highly redundant, one first reduces the dimensionality by performing principal component analysis (PCA) to keep the first 100 principal components.

![Figure 5.3: Metric learning using the Iris plant data set. (a) 4-D data set projected onto a 2-D space; data set after metric learning based on (b) LMNN and (c) LSML.](image)

The 3-nearest neighbor classification results based on different metrics are summarized in Table 5.2 below. For each metric and training set size, it shows the mean classification rate and standard deviation over 10 random runs corresponding to different randomly generated training sets. It can be seen that both LMNN and LSML are significantly better than the Euclidean metric, with LSML being slightly better.

**Table 5.2: The three different distance metrics.**

<table>
<thead>
<tr>
<th>% Labeled Data</th>
<th>Minist</th>
<th>Isolet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
<td>20%</td>
</tr>
<tr>
<td><strong>Euclidean</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>0.3770 (± 0.030)</td>
<td>0.5095 (± 0.006)</td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LMNN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>0.8711 (± 0.009)</td>
<td>0.9270 (± 0.004)</td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LSML</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td>0.5959 (± 0.010)</td>
<td>0.9315 (± 0.014)</td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.4: Pictorial Representations of Table 5.1.
5.3.2. Efficiency

Besides the promising performance of LSML for different semi-supervised learning tasks, our method also has the additional advantage of being efficient. In the experiments performed, LSML is much faster than LMNN which solves an SDP problem based on gradient descent, and LLMA, which optimizes a non-convex objective function using an iterative algorithm.

Let us denote a set of \(n\) data points in a \(d\)-dimensional input space by \(X = \{x_1, x_2, \ldots, x_m\}\). As in [Bar-Hillel, 2003], one can only consider pair-wise similarity constraints which are given in the form of a set \(S_0\) of similar point pairs. Intuitively, one can want to transform the \(n\) data points to a new space in which the points in each similar pair will get closer to each other. To preserve the topological relationships between data points, one moves not only the points involved in the similar pairs but also other points. For computational efficiency, one can resort to linear transformation. One promising approach is to apply locally linear transformation so that the overall transformation of all points in \(X\) is linear locally but nonlinear globally, generalizing previous metric learning methods based on applying linear transformation globally [Bar-Hillel et al., 2003] [Xing E., 2003]. One can call this new metric learning method locally linear metric adaptation (LLMA). However, caution should be taken when applying linear transformation to reduce the distance between similar points, as a degenerate transformation will simply map all points to the same location so that all inter-point distances vanish (and hence become the smallest possible). Obviously this degenerate case is undesirable and should be avoided.

Relevant Component Analysis (RCA) is a method that seeks to identify and down-scale global unwanted variability within the data. The method changes the feature space used for data representation, by a global linear transformation which assigns large weights to "relevant dimensions" and low weights to "irrelevant dimensions" (cf. (Tenenbaum & Freeman, 2000)). These "relevant dimensions" are estimated using chunklets.

One can define a chunklet as a subset of points that are known to belong to the same although unknown class; chunklets are obtained from equivalence relations by applying a transitive closure. The RCA transformation is intended to reduce clutter, so that in the new
feature space, the inherent structure of the data can be more easily unraveled. The method can be used as a preprocessing step for the unsupervised clustering of the data or nearest neighbor classification.

5.4 Experiments on Image Retrieval

5.4.1 Image Databases and Feature Representation

The image retrieval experiments are based on two image databases. One database is a subset of the Corel Photo Gallery, which contains 1010 images belonging to 10 different classes. The 10 classes include bear (122), butterfly (109), cactus (58), dog (101), eagle (116), elephant (105), horse (110), penguin (76), rose (98), and tiger (115). Another database contains 547 images belonging to six classes that one downloaded from the Internet. The image classes are manually defined based on high-level semantics.

5.4.2. Experimental Settings

The similarity constraints used in LLMA and LSML are obtained from the relevance feedback of the CBIR system, with each relevant image and the query image forming a similar image pair. It is straightforward to construct chunklets for RCA from the similarity constraints.

Besides RCA and LLMA, one can also compare the image retrieval performance of our method with the baseline method of using Euclidean distance without distance learning. In summary, the following five methods are included in this comparative study: (1) Euclidean distance without metric learning; (2) Global metric learning with RCA; (3) Nonmetric distance boosting with DistBoost; (4) Local metric learning with LLMA; (5) Local metric learning with modified LSML.

One can measure the retrieval performance based on cumulative neighbor purity curves. Cumulative neighbor purity measures the percentage of correctly retrieved images in the k nearest neighbors of the query image, averaged over all queries, with k up to some value K (K = 20 or 40 in our experiments).

For each retrieval task, one can compute the average performance statistics over 5 randomly generated sets of similar image pairs. For both databases, the number of similar image pairs is set to 150, which is about 0.3% and 0.6%, respectively, of the total number of possible image pairs in the databases.
5.4.3. Experimental Results

Figure 5.5 shows the retrieval results on the first image database based on cumulative neighbor purity curves.

One can see that local metric learning with LLMA and Modified LSML methods significantly improves the retrieval performance, while LSML leads to slightly better result than LLMA. DistBoost obtains high recall with more images retrieved, but low precision on the top ranking images, which most online image retrieval engines return. The retrieval results on the second image database are shown in Figure 5.6. Again, LSML outperforms other metric learning methods.
5.5 Conclusion

The modified metric learning method possesses the simultaneous advantages of being flexible, efficient semi-supervised learning tasks. In particular, this method can be used to boost image retrieval performance.