# Parameter Free Large Margin Nearest Neighbor for Distance Metric Learning

**Kun Song,[†] Feiping Nie,[‡*] Junwei Han,[†*] Xuelong Li[§]**

[†]School of Automation, Northwestern Polytechnical University, Xi'an, 710072, Shaanxi, P. R. China

[‡]School of Computer Science and Center for OPTIMAL, Northwestern Polytechnical University, Xi'an,710072, P. R. China

[§] Center for OPTIMAL, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, Shaanxi, P. R. China.

{songkun123000, feipingnie, junweihan2010}@gmail.com, xuelong_li@ieee.org

## Abstract

We introduce a novel supervised metric learning algorithm named parameter free large margin nearest neighbor (PFLMNN) which can be seen as an improvement of the classical large margin nearest neighbor (LMNN) algorithm. The contributions of our work consist of two aspects. First, our method discards the cost term which shrinks the distances between inquiry input and its $k$ target neighbors (the $k$ nearest neighbors with same labels as inquiry input) in LMNN, and only focuses on improving the action to push the imposters (the samples with different labels form the inquiry input) apart out of the neighborhood of inquiry. As a result, our method does not have the parameter needed to tune on the validating set, which makes it more convenient to use. Second, by leveraging the geometry information of the imposters, we construct a novel cost function to penalize the small distances between each inquiry and its imposters. Different from LMNN considering every imposter located in the neighborhood of each inquiry, our method only takes care of the nearest imposters. Because when the nearest imposter is pushed out of the neighborhood of its inquiry, other imposters would be all out. In this way, the constraints in our model are much less than that of LMNN, which makes our method much easier to find the optimal distance metric. Consequently, our method not only learns a better distance metric than LMNN, but also runs faster than LMNN. Extensive experiments on different data sets with various sizes and difficulties are conducted, and the results have shown that, compared with LMNN, PFLMNN achieves better classification results.

## Instruction

Distance metric learningis an important topic in the field of machine learning (Xing et al. 2003; Yang and Sukthankar 2006; Parameswaran and Weinberger 2010; Wang, Nie, and Huang 2014; Xiang, Nie, and Zhang 2008; You et al. 2016; Guillaumin and Verbeek 2009; Weinberger, Blitzer, and Saul 2005). The largest margin nearest neighbor (LMNN) algorithm (Weinberger, Blitzer, and Saul 2005) is one of the most popular metric learning methods. Several reasons account for lots of attentions paid on LMNN: firstly, LMNN always yields competitive results for many classification tasks because it captures the local information of the data set (Weinberger and Saul 2009; Ma, Crawford, and Tian 2010; Mensink et al. 2012; You et al. 2016); secondly, LMNN with similarity to support vector machines (SVMs) is much easily extended to non-linear versions by adopting the kernel technique (Do et al. 2012; Torresani and Lee 2006), which expands the application domain of LMNN; thirdly, the linear transform matrix $L$ learned by LMNN supervisedly contains many statistical information of training set, thus it can be applied into many machine learning fields, such as feature extraction, multi-task classification and dimensionality reduction (Kedem et al. 2012; Torresani and Lee 2006; Parameswaran and Weinberger 2010).

LMNN is inspired by improving the performance of kNN algorithm whose performance depends crucially on the distance metric. LMNN learns a linear transform matrix $L$ to construct the Mahalanobis distance by utilizing the local information of the training set. Under such distance metric, the $k$ nearest neighbors with same labels (target neighbors) of an inqury $\vec{x}_i$ are pulled nearer together and the neighbors with different labels (imposters) are pushed out of the neighborhood. In the model of LMNN, such two competitive actions on the samples are implemented by constructing two cost terms which are weighted by a parameter needed to be tuned previously. However, it is difficult to find the optimal distance metric effectively by minimizing the combination of such two competitive cost terms. Actually, the model of the LMNN is very complex, which not only considers the imposters of an inquiry sample but also concerns the target neighbors. Such imposters and target neighbors can be regarded as the constraints of the model. The more the constraints are, the more difficult the way to find the solution is. Actually, it is easy to find that, the missions of the actions of pulling and pushing are to make the inquiry's neighborhood 'pure'. To complete such mission, the action pushing the imposters apart form the neighborhood is enough.

In this paper, we propose a novel metric learning algorithm named parameter free large margin nearest neighbor (PFLMNN). Different from LMMN pulling target neighbors together and pushing imposters apart at the same time, our approach only considers the action to pushing the imposters

*Corresponding authors.

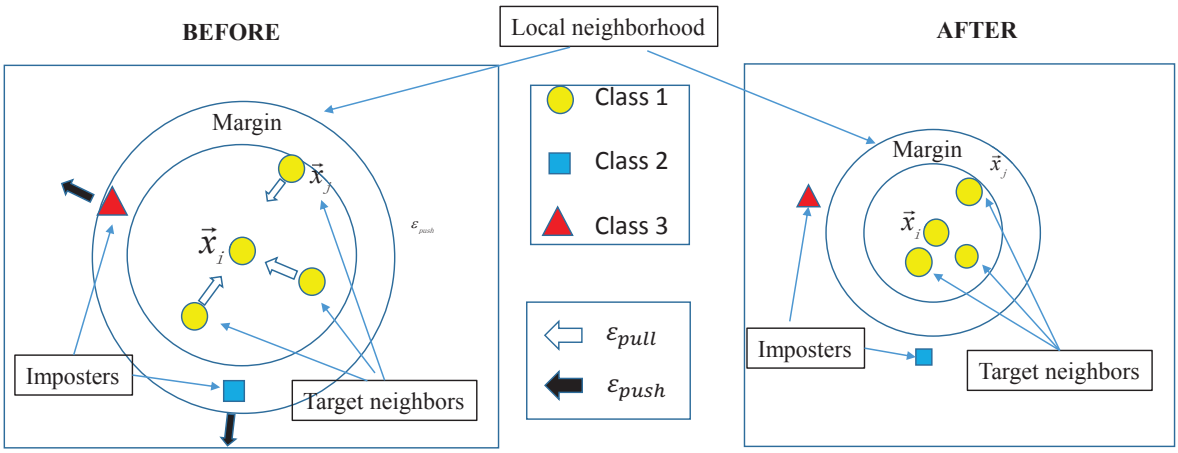


Figure 1: Illustration of how LMNN affects an input's neighborhood. The $k = 3$ neighbors are pull together in the small radius while the differently labeled inputs are pull out of the small radius by some finite margins.

out of the neighborhood. In this way, we simplify the mission of our optimization problem, compared with LMNN. To enhance the power of penalizing large distance between $k$ nearest neighbors and inquiries, we develop a novel method to push the imposters, which utilizes the geometry information between the imposters ignored by LMNN. In brief, only the nearest active imposter (one of the $k$ nearest neighbors of the inquiry, however, does not share the same label with inquiry) of each inquiry is considered. Under a distance metric, when the nearest imposter of an inquiry is out of the neighborhood, all other imposters are out. In this way, the constraints shown by the imposters needed to be pushed are reduced, so that the proposed model would be much simpler without weakening its constrained power. Consequently, our model is easier to be optimized compared with LMNN. In addition, no parameter needs to be tuned in the cost function because of only one term owned by our model, which makes the proposed method more convenient to use. Extensive experiments are conducted to demonstrate the effective of our method.

## Revist LMNN

In this section, we briefly revisit the notations of LMNN which would be used in our model. Let $\{\vec{x}_i, y_i\}_{i=1}^n$ denote a training set of $n$ labeled examples with inputs $\vec{x}_i \in R^d$ and discrete class labels $y_i \in \{1, 2, \cdots, c\}$, where $c$ is the class number. The goal of LMNN is to learn a linear transformation $\mathcal{L} : R^d \to R^d$, which is used to define the Mahalanobis distance as:

$$\mathcal{D}(\vec{x_i}, \vec{x_j}) = \|L(\vec{x_i} - \vec{x_j})\|_2 = \sqrt{(\vec{x_i} - \vec{x_j})^T M (\vec{x_i} - \vec{x_j})} \quad (1)$$

where $M = L^T L$ is a symmetric positive definite matrix ($M \succeq 0$).

For an inquiry input $\vec{x_i}$, we call its $k$ nearest neighbors with the same label of $\vec{x_i}$ as "target neighbors", and all the samples in the data set with different labels from $y_i$ as "imposters". LMNN aims at learning a Mahalanobis distance metric that keeps each input $\vec{x_i}$ closer to its target neighbors than the imposters by one distance unit, i.e. large margin. The relation of $\vec{x_i}$ 's target neighbor $\vec{x_j}$ and imposter $\vec{x_l}$ can be expressed as linear inequality constraint with respect to the $\mathcal{D}(\cdot, \cdot)$:

$$\mathcal{D}^2(\vec{x_i}, \vec{x_l}) - \mathcal{D}^2(\vec{x_i}, \vec{x_j}) \geq 1 \quad (2)$$

To achieve this goal, LMNN constructs two cost terms in its loss function. The first term penalizes large distances between each input $\vec{x_i}$ and its target neighbors $\vec{x_j}$, and the second term penalizes small distance between each input and its imposters. In this way, LMNN pulls the "target neighbors" of each input example closer together, and pushes differently labeled examples further apart. We use $S_i$ to denote the set that contains "target neighbors" of $\vec{x_i}$ and $P_i$ to denote the set that consists of the corresponding imposters of $\vec{x_i}$ . The loss function of LMNN is defined as

$$\begin{aligned} \epsilon(M) =& (1-\lambda) \sum_i \sum_{j \in S_i} \mathcal{D}^2(\vec{x_i}, \vec{x_j}) \\ &+ \lambda \sum_i \sum_{j \in S_i, l \in P_i} [1 + \mathcal{D}^2(\vec{x_i}, \vec{x_j}) - \mathcal{D}^2(\vec{x_i}, \vec{x_l})]_+ \end{aligned} \quad (3)$$

where in the second term $[z]_+ = max(z, 0)$ denotes the standard hinge loss and $\lambda > 0$ is a positive constant. We call the first term in Eq.(3) as $\epsilon_{pull}$ and the second term as $\epsilon_{push}$. It is easy to find that such two terms have competing effect, since the first is reduced by shrinking the distances between examples and the second is reduced by magnifying them. $\lambda$ palys the role of balancing the effect of the two terms. Figure 1 presents the illustration.

Eq.(3) can be transformed to an instance of semi-definite programming (SDP) by introducing slack variables $\xi_{ijl}$ as in Table 1. It indicates the model of LMNN is convex (Vandenberghe and Boyd 1996). Suppose that the amount of examples in each class is equal, the number of constraints

Table 1: The model of SDP for LMNN.

| |
|---|
| ***Minimize*** $(1-\lambda)\sum_{i,j\in S_i}(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j})$ $+\lambda\sum_{i,j\in S_i,l\in P_i}\eta_{ij}(1-y_{il})\xi_{ijl}$ |
| **subject to** |
| $(1)(\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l})-(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j})\geq 1-\xi_{ijl}$, |
| $(2)\forall j\in S_i, \forall l\in P_i, \xi_{ijl}\geq 0$,, |
| $(3)M\succeq 0$. |

including (1) (2) of SDP described in Table 1 is about $2kn^2(1-1/c)$, thus the standard off-the-shelf packages are not suitable when the data set is huge.

As seen from Eq.(3), we can find that both of the two terms try to achieve the goal stated in Eq.(2). However, to achieve this goal, only the second term is enough, i.e, if we remove the term $\epsilon_{pull}$ from Eq.(3), the goal in Eq.(2) would be also achieved. In fact, for an optimization problem, the more complex the model is, the harder the solution to be found is. If the first term $\epsilon_{pull}$ is added in the objective, it means some extra constraints on the optimization problem are added. In mathematics, adding such extra constraints would make the searching space smaller. The result found from the smaller searching space may have poor performance.

# Parameter Free Large Margin Nearest Neighbor

## Problem Formulation

Compared with LMNN, PFLMNN algorithm only focuses on the action that penalizes the small distance between the imposters and each inquiry input. As the compensation of the abandon of the action to pull target neighbors near together compared with LMNN, it utilizes the geometry information of the samples in the feature space. For an inquiry $\vec{x_i}$ and its target neighbor $\vec{x_j}$, if the nearest imposter $\vec{x_l}$ satisfies the inequality in Eq.(2), all other imposters would follow such inequality relation too. Such relationship between the imposters is called geometry information. The optimization problem of PFLMNN is defined as follows.

$$\min_{M\succeq 0}\sum_i\sum_{j\in S_i}[1-\min_{l\in P_i}((\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l}) - (\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j}))]_+ \tag{4}$$

where the $S_i$ is the set which contains $k$-nearest target neighbors of $\vec{x_i}$ and $P_i$ is the set which contains all the instances with different labels (imposters) from $\vec{x_i}$. As seen in (4), there is only one cost term compared with LMNN, so it requires no parameter to be tuned. The sub-optimization problem in Eq.(4) is used to find the nearest imposter.

Why the sub-optimization problem can improve the performance of the proposed method compared with LMNN? The reasons consist of two aspects. Firstly, in LMNN, the second term is the sum of distances of each inquiry and its imposters. Each sample in this term is treated equally, thus the imposters with different distances from the inquiry may have negative impact on each other. As a result, the performance for pulling all the active imposters apart would be reduced. Secondly, each imposter which triggers the hinge loss function can be regarded as a constraint of the model. Equally treating the imposters would make the amount of the constraints be huge, which would increase the difficulty of finding optimal solution.

## Relation with SDP

In this section, we explore the relationship of the PFLMNN with SDP, which would contribute us to solve the proposed method. As seen from Eq.(4), the optimization problem of PFLMNN can be seen as combination of two sub-problems. The first one aims to find the nearest imposters of $\vec{x_i}$ with respect to target neighbor $\vec{x_j}$ under $M$, and the second one optimizes metric matrix $M$ by using those nearest imposters. The two sub-optimization problems are inseparably intertwined, so it is difficult to solve it in a straight-forward form. Now, we transform the Eq. (4) to a semi-definite programming problem which is a convex problem.

Since the hinge loss can be "mimicked" by introducing slack variables $\xi_{ij}$ (Weinberger, Blitzer, and Saul 2005), the optimization problem in Eq. (4) can be transformed to following optimization problem.

$$\begin{aligned} \underset{M}{\textbf{\textit{minimize}}} \quad & \sum_i\sum_{j\in S_i}\xi_{ij} \\ \textbf{\textit{subject to}} \quad & \min_{l\in P_i}((\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l}) \\ & -(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j}))\geq 1-\xi_{ij}, \\ & M\succeq 0, \\ & \xi_{ij}\geq 0. \end{aligned} \tag{5}$$

For the first constraint in the optimization problem stated in Eq.(5), there is relationship as follows.

$$\min_{l\in P_i}((\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l})-(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j}))\geq 1-\xi_{ij}$$

$$\Leftrightarrow \forall l\in P_i, 1-\xi_{ij}\leq(\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l})-(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j})$$

Thus, the Eq. (4) can be further transformed into a instance of standard SDP described in Table 2.

$M\succeq 0$ indicates that matrix $M$ is required to be positive semi-definite, and the linear transform matrix $L$ can be calculated by matrix decomposition of $M$.

Suppose each class has the same number of instances, the numbers of samples in set $P_i$ and set $S_i$ are $n(1-1/c)$ and $k$ respectively. So the amount of constraints (1) and (2) in the Table 2 is $kn^2(1-1/c)+kn$, which is less than that of LMNN ($2kn^2(1-1/c)$). It is proved that, the proposed model is quite simpler than LMNN.

Similar to the optimization problem of LMNN, our model is a convex problem which can be solved by standard SDP programmer. However, most off-the-shelf packages fail to solve this problem due to the expensive requirement of the physical memory.

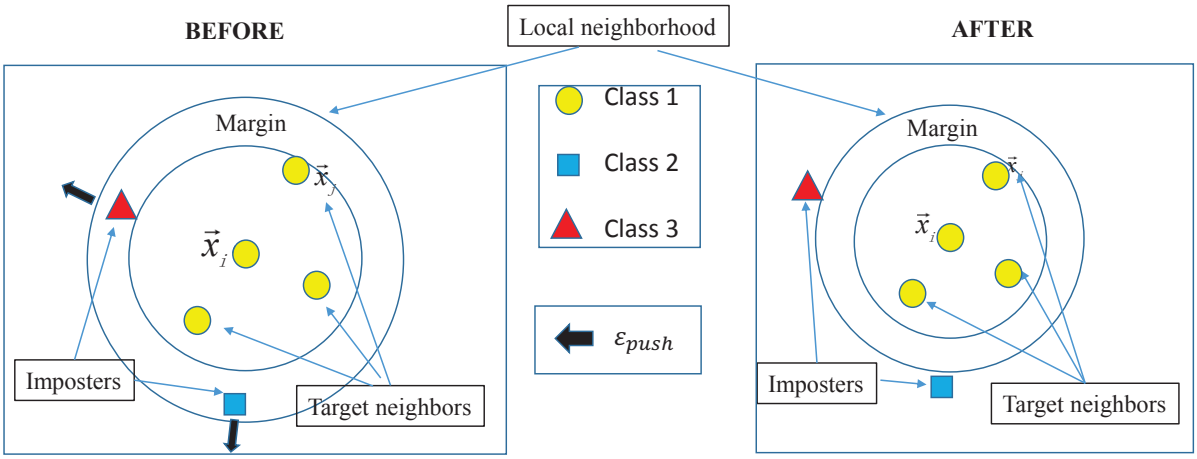


Figure 2: Illustration of PFLMNN. Different from LMNN, PFLMNN only takes the action to pull the differently labeled inputs out of the small radius by some finite margins.

Table 2: SDP model of PFLMNN.

| |
|---|
| ***Minimize*** $\sum_i \sum_{j\in S_i} \xi_{ij}$ |
| **subject to** |
| (1) $\forall l \in P_i, (\vec{x_i}-\vec{x_l})^T M(\vec{x_i}-\vec{x_l})-(\vec{x_i}-\vec{x_j})^T M(\vec{x_i}-\vec{x_j}) \geq 1-\xi_{ij}$, |
| (2) $\forall j \in S_i, \xi_{ij} \geq 0$, |
| (3) $M \succeq 0$. |

## Optimization

In the previous section, we have demonstrated the convexity of the proposed problem. The proposed method can be solved by a sub-gradient descent method. Before giving the solution, we introduce an useful theorem.

**Theorem 1:** Suppose $\tilde{M} \in R^{n\times n}$ is a symmetric matrix diagonal matrix, $S_+^n$ is the set of positive semi-definite matrices with size $n \times n$. The projection of $\tilde{M}$ in $S_+$ is denoted $M$. There is

$$M = U\Sigma_+ U^T \tag{6}$$

where $U$ is a orthogonal unit matrix which makes $\Sigma = U^T\tilde{M}U$ diagonal, and $\Sigma_+ = max\{\Sigma, 0\}$.

The proof of Theorem 1 can be found in (Han 1998). Following it, we can project an symmetric matrix into a semi-definite cone.

Now, we introduce the sub-gradient method for solving PFLMNN. Suppose $X_{i,j} = (x_i - x_j)(x_i - x_j)^T$, there is $Tr(MX_{ij}) = (x_i - x_j)^T M(x_i - x_j)$, where $Tr(\cdot)$ is the trace of the matrix $\cdot$. Let $\Gamma(M)$ denote the objective function of Eq. (4), there is

$$\Gamma(M) = \sum_{i,j\in S_i} [1 - \min_{l\in P_i} Tr(MX_{il}) + Tr(MX_{ij})]_+ \tag{7}$$

The sub-gradient of $\Gamma(M)$ respect to $M$ is given as follow.

$$\begin{aligned} \frac{\partial\Gamma(M)}{\partial M} &= \sum_{i,j\in S_i} [\varepsilon_{ijl_m}]_+ (X_{ij} - X_{il_m}) \\ l_m &= \operatorname*{argmin}_l (\vec{x_i} - \vec{x_l})^T M(\vec{x_i} - \vec{x_l}) \\ \varepsilon_{ijl_m} &= 1 - Tr(MX_{il_m}) + Tr(MX_{ij}) \end{aligned} \tag{8}$$

where $[\varepsilon_{ijl_m}]_+ = 1$, if $\varepsilon_{ijl_m} > 0$, else $[\varepsilon_{ijl_m}]_+ = 0$.

We let $M_i$ denotes the feasible point at the $i$-th step. The point $\tilde{M}_{i+1} = M_i - \gamma\frac{\partial\Gamma(M)}{\partial M}$. Obviously, $\tilde{M}_{i+1}$ is a symmetric matrix, but we can not grantee it as semi-definite. We should project $\tilde{M}_{i+1}$ into the feasible region according to Theorem 1. The details of the sub-gradient method are presented in Algorithm 1.

**Algorithm 1 PFLMNN**

**Input:** Data sets $\{\vec{x}_i, y_i\}_{i=1}^n$.
**Initialize** $M_0$.
**Repeat**

1. Calculate sub-gradient $\nabla G_i = \frac{\partial\Gamma(M)}{\partial M}$ at $M_i$ by Eq. (8);
2. Calculate $\tilde{M}_{i+1} = M_i + \lambda\nabla G_i$;
3. Do eigenvalue decomposition on $\tilde{M}_{i+1}$ and obtain $U$, $\Sigma_+$;
4. $M_{i+1} = U^T\Sigma_+ U$, and $L_{i+1} = sqrt(\Sigma_+)U$

**Until** Convergence
**Output:** $L$,$M$;

## Kernel Parameter Free Large Margin Nearest Neighbor

In this section, we discuss how to 'kernelize' the proposed method in order to learn the metric under non-linear environment.

We firstly calculate the sub-gradient of objective function

in Eq. (4) with respect to $L$. By substituting $M = L^T L$ in the objective function in Eq. (4), the sub-gradient is calculated as.

$$\begin{aligned}\frac{\partial \Gamma(L)}{\partial L} =& \sum_{i,j} [\varepsilon_{ijl_m}]_+ 2L((\vec{x_i} - \vec{x_j})(\vec{x_i} - \vec{x_j})^T \\ & - (\vec{x_i} - \vec{x_{l_m}})(\vec{x_i} - \vec{x_{l_m}})^T) \\ l_m =& \underset{l}{\text{argmin}} (\vec{x_i} - \vec{x_l})^T L^T L (\vec{x_i} - \vec{x_l})\end{aligned} \tag{9}$$

where,
$\varepsilon_{i,j,l_m} = 1 - (\vec{x_i} - \vec{x}_{l_m})^T L^T L (\vec{x_i} - \vec{x}_{l_m}) + (\vec{x_i} - \vec{x_j})^T L^T L (\vec{x_i} - \vec{x_j})$, and if $\varepsilon_{ijl_m} > 0$, $[\varepsilon_{ijl_m}]_+ = 1$, otherwise, $[\varepsilon_{ijl_m}]_+ = 0$.

Suppose $\phi$ is a nonlinear map function, it projects $x_i$ to a high dimensional feature space by $\phi_i = \phi(\vec{x_i})$. There exists a kernel function $\kappa$ that can be used to compute the feature inner products without carrying out the map, i.e. $\kappa(\vec{x_i}, \vec{x_j}) = \phi_i^T \phi_j$. We modify our objective $\Gamma(L)$ by substituting inputs $\vec{x_i}$ with mapped features $\phi(\vec{x_i})$ into Eq. (9) and obtain

$$\begin{aligned}\Gamma(L) =& [1 - \min_{l \in P_i} ((\phi_i - \phi_l)^T L^T L (\phi_i - \phi_l) \\ & - (\phi_i - \phi_j)^T L^T L (\phi_i - \phi_j))]_+\end{aligned} \tag{10}$$

So the gradient in the feature space can now be written as:

$$\begin{aligned}\frac{\partial \Gamma(L)}{\partial L} =& 2L \sum_{i,j} [\varepsilon_{ijl_m}]((\phi_i - \phi_j)(\phi_i - \phi_j)^T \\ & - (\phi_i - \phi_{l_m})(\phi_i - \phi_{l_m})^T)\end{aligned} \tag{11}$$

where $l_m = argmin_l (\phi_i - \phi_l)^T L^T L (\phi_i - \phi_l)$, $\varepsilon_{i,j,l_m} = 1 - (\phi_i - \phi_{l_m})^T L^T L (\phi_i - \phi_{l_m}) + (\phi_i - \phi_j)^T L^T L (\phi_i - \phi_j)$.

Let $\Phi = [\phi_1, \cdots, \phi_n]^T$, we establish linear equation $L = \Omega\Phi$, where $\Omega$ is the matrix allowing us to write $L$ as linear combination of feature points. This form of nonlinear map is analogous to that used in kernel-PCA and it allows to parameterize the transformation $L$ in terms of only $d \times n$ parameters, the entries of the matrix $\Omega$. We now introduce the following Lemma which we will later use to derive an iterative update rule for $L$.

**Lemma 1** The gradient in feature space can be computed as $\frac{\partial \Gamma(L)}{\partial L} = \Psi\Phi$, where $\Psi$ depends on features $\phi_i$, solely in terms of dot product $\phi_i^T \phi_j$.

**Proof:** Defining $k_i = \Phi\phi_i = [\kappa(x_i, x_i), \cdots, \kappa(x_n, x_i)]$, non-linear feature projections can be computed as $L\phi_i = \Omega\Phi\phi_i = \Omega k_i$. From this we derive:

$$\begin{aligned}\frac{\partial \Gamma(L)}{\partial L} =& 2c\Omega \sum_{i,j} [\varepsilon_{ijl_m}]_+ ((k_i - k_j)(\phi_i - \phi_j)^T \\ & - (k_i - k_{l_m})(\phi_i - \phi_{l_m})^T) \\ =& \Psi\Phi\end{aligned}$$

where

$$\Psi = 2\Omega \sum_{i,j} [\varepsilon_{ijl_m}]_+ (E_i^{k_i - k_j} - E_j^{k_i - k_j} - E_i^{k_i - k_{l_m}} + E_{l_m}^{k_i - k_{l_m}})$$

$E_i^{\vec{v}} = [0, \cdots, 0, \vec{v}, 0, \cdots, 0]$ is the $n \times n$ matrix having vector $\vec{v}$ in the $i$-th column and all 0 in the other columns. □

The key idea is to iteratively update $\Omega$ rather than $L$. For example, using gradient descent as optimization we derive update rule:

$$L_{new} = L_{old} - \lambda \frac{\partial \Gamma(L)}{\partial L}|_{L=L_{old}} = (\Omega_{old} - \lambda\Psi_{old})\Phi = \Omega_{new}\Phi \tag{12}$$

where $\lambda$ is the learning rate. We carry out this optimization by iterating the update $\Omega \leftarrow (\Omega - \lambda\Psi)$ until convergence. For classification, we project points onto the learned space by exploiting the kernel trick: $L\phi_q = \Omega k_q$.

Table 3: The details of five data sets.

| data set | # classes | # examples | # dimension | |
|---|---|---|---|---|
| | | | Original | Processed |
| Iris | 3 | 150 | 4 | 4 |
| Wine | 3 | 178 | 13 | 13 |
| Isolet | 26 | 6238 | 617 | 172 |
| AT&T | 40 | 400 | 1170 | 30 |
| Coil-100 | 100 | 7200 | 1024 | 103 |
| USPS | 10 | 11000 | 256 | 96 |

## Numerical Experiment

In this section, we evaluate the performance of PFLMNN and its kernelized version on several data sets. The classification accuracy is adopted as the metric.

### Data set description

All the experiments are conducted on six data sets with different sizes and difficulties, i.e. Iris, Wine, Isolet, AT&T[1], Coil100 (Nene, Nayar, and Hiroshi 1996) and USPS (Hull 1994). Among those data sets, the Wine, Iris and Isolet are taken from the UCI Machine learning Repository[2]. AT&T, Coil100, USPS are the hunman face image data sets, objective image data set and the hand-writing digit data, respectively. All the six data sets are often adopted as benchmark data sets for distance metric learning in recent works. Since the dimensionality of some data sets is very high, principal component analysis (PCA) is used for feature reduction (with score 95%), both to speed up the model training and avoid overfitting. The details of the those data sets are shown in Table 3.

### Experiment setting

We conduct two series of experiments. Firstly, we consider the PFLMNN not using the kernelized technique. We compare it with several state-of-the-art supervised distance metric learning methods, i.e. large margin nearest neighbor algorithm (LMNN) and sparse compositional metric learning (SCML) (Yuan, Bellet, and Fei 2014), local distance metric learning (LDML) (Yang and Sukthankar 2006), information theory metric learning(ITML) (Davis, Kulis, and Jain 2007), and regressive virtual metric learning

[1] http://www.uk.research.att.com/facedatabase.html

[2] http://www.ics.ucl.edu/ mlean/MLRepository.html

Table 4: Comparison of our approach PFLMNN with several baselines in the linear case.

| Base | kNN | LMNN | ITML | LDML | SCML | RVML | FLMNN |
|---|---|---|---|---|---|---|---|
| Iris | 92.12±2.31 | 94.23±1.54 | 93.42±2.52 | 92.15±2.47 | 94.32±2.48 | 93.43±1.34 | **96.41±1.23** |
| Wine | 94.18±1.59 | 98.36±1.03 | 97.42±1.21 | 95.56±1.79 | 96.91±1.93 | 97.82±1.45 | **98.55±1.67** |
| Isolet | 88.97±2.41 | 95.83±3.21 | 94.83±2.67 | 93.42±1.49 | 89.61±2.37 | 91.40±3.41 | **97.45±1.21** |
| Letter | 94.74±1.27 | 96.43±1.21 | 95.43±1.28 | 95.57±1.12 | 961.3±1.45 | 90.25±1.61 | **97.92±1.54** |
| Coil-100 | 94.63±1.34 | 96.84±1.56 | 96.21±2.32 | 95.32±2.46 | 95.44±1.54 | 95.55±1.51 | **98.89±1.33** |
| USPS | 86.32±1.32 | 98.28±1.25 | 96.43±1.21 | 97.32±2.21 | 97.46±1.32 | 98.21±1.21 | **99.42±1.22** |
| AT&T | 91.13±3.22 | 96.81±1.41 | 97.12±2.21 | 96.45±2.11 | 96.52±1.63 | 96.42±1.34 | **97.21±2.34** |

Table 5: Comparison of our approach KPFLMNN with several baselines in the non-linear case.

| Base | LMNN-KPCA | ITML-KPCA | LDML-KPCA | GBLMNN | SCMLLOCAL | KRVML | KPFLMNN |
|---|---|---|---|---|---|---|---|
| Iris | 95.21±1.43 | 94.56±2.11 | 93.24±1.43 | 96.24±1.26 | 94.31±1.21 | 94.32±2.32 | **97.23±2.12** |
| Wine | 95.82±1.29 | 97.43±1.31 | 92.18±2.12 | 98.00±1.21 | 96.55±2.11 | 96.82±2.24 | **98.91±1.73** |
| Isolet | 96.28±2.31 | 95.44 ±1.76 | 88.57±2.11 | 96.02 ±1.92 | 91.40±2.03 | 95.96±1.11 | **97.24±1.21** |
| Letter | 97.17±1.58 | 96.35±1.93 | 95.39±1.47 | 96.51±2.34 | 96.63±2.13 | 91.26±1.59 | **98.43±1.47** |
| Coil-100 | 96.42 ±1.78 | 96.31±1.73 | 95.43±2.12 | 97.52±1.26 | 96.21 ±1.21 | 94.92±2.31 | **97.43±1.12** |
| USPS | 98.56±0.43 | 97.43±0.82 | 87.42±2.96 | 98.72±0.43 | 97.13 ±1.03 | 97.92±0.52 | **98.62±0.46** |
| AT&T | 97.21 ±2.21 | 97.51±1.75 | 92.32±2.14 | 98.12 ±1.76 | 96.42±1.82 | 96.82±1.12 | **98.51±1.31** |

(RVML)(Perrot and A. 2015). In addition, we also compare FPLMNN with $k$-nearest neighbor classification without metric learning.

In the second series, we consider the kernelized PFLMNN (KPFLMNN) which adopts kernelized technique to improve its performance. Since LMNN, ITML, IDML, are not kernelized, we adopt kernelized PCA as a pro-progress to transform them to kenerlized methods. Some kernelized metric learning methods, i.e. GBLMNN (Kedem 2012), a non-linear version of LMNN and KRVML (Perrot and A. 2015), the kernelized version of RVML, are considered. We also report the results of SCMLOCAL which is the local version of SCML.

In all of the experiments reported here, the parameter $\lambda$ of LMNN is tuned by 5-fold cross validation (For the purpose of cross-validation, the training sets would be partitioned into training and validation sets at 80/20), and the searching grid is set at $\{0.02, 0.04, \cdots, 1\}$. The nearest neighbors' number, i.e. $k$ is set by cross-validation as recommended in (Weinberger and Saul 2009) for all the methods in our experiment. For KPFLMNN, KPCA, KRVML, the Gaussian RBF kernel is adopted, and the variance of the RBF kernel is set as the mean of Euclidean distances between all pairwise samples in the training set. All of the experiment results are averaged over several runs of randomly generated 70/30 splits of the data. Each experiment runs 50 times independently.

## Experiment results

The classification results for PFLMNN and KPLMNN are presented in Table 4 and Table 5. As seen from the Table 4, the proposed method, i.e. PFLMNN has achieved the best performance compared with other methods, which is highlighted by bold words. There is the same conclusion in the Table 5. In Table 5, KPLMNN achieves the best performance. Those results can prove the effective of the proposed methods.

## Conclusion

In this paper, we have proposed a novel large margin nearest neighbor algorithm, i.e. PFLMNN. Different from LMNN, our algorithm only considered the action of pushing the imposters apart from the neighborhood, so that there is no parameter needed to be tuned in our model, which is more convenient to use in practice. To compensate the abandon of the cost term for penalizing large distances between target neighbors, we developed a novel framework which utilizes the geometry information of imposters, to make the action to push the imposters more effectively. As a result, there are less constraints considered in the corresponding SDP transformation, compared with LMNN. We evaluated our algorithm on several data sets with various sizes and difficulties. Compared with state-of-the-art methods, PFLMNN achieved comparative classification results. In the future works, we would apply our method to the application such as computer version(Lu and Yuan 2013; Lu and Wu 2014; Han et al. 2015; Cheng et al. 2015).

## Acknowledgments

This work was supported in part by the National Science Foundation of China under Grant 61473231 and Grant 61522207.

## References


Cheng, G.; Han, J.; Guo, L.; Liu, Z.; Bu, S.; and Ren, J. 2015. Effective and efficient midlevel visual elements-oriented land-use classification using VHR remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing* 53(8):1–12.

Davis, J.; Kulis, B.; and Jain, P. 2007. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning. ACM*, 209–216.

Do, H.; Kalousis, A.; Wang, J.; and Woznica, A. 2012. A metric learning perspective of svm: on the relation of svm and lmnn. *arXiv preprint arXiv:1201.4714*.

Guillaumin, M., and Verbeek, J. 2009. Is that you? metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th international conference on*, 498–505. IEEE.

Han, J.; Zhang, D.; Hu, X.; Guo, L.; Ren, J.; and Wu, F. 2015. Background prior-based salient object detection via deep reconstruction residual. *IEEE Transactions on Circuits and Systems for Video Technology* 25(8):1309–1321.

Han, Q. 1998. Projection and contraction methods for semidefinite programming. *Applied Mathematics and Computation* 95(2):275–289.

Hull, J. J. 1994. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 16(5):550–554.

Kedem, D.; Tyree, S.; Sha, F.; Lanckriet, Gert, R.; and Weinberger, Kilian, Q. 2012. Non-linear metric learning. In *Advances in Neural Information Processing Systems*, 2573–2581.

Kedem, D.; Tyree, S. S. F. 2012. Non-linear metric learning. In *Advances in Neural Information Processing Systems*. 2573–2581.

Lu, X., and Wu, H. 2014. Double constrained nmf for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing* 52(5):2746–2758.

Lu, X., and Yuan. 2013. Image super-resolution via double sparsity regularized manifold learning. *IEEE Transactions on Circuits and Systems for Video Technology* 23(12):2022–2033.

Ma, L.; Crawford, Melba, M.; and Tian, J. 2010. Local manifold learning based nearest neighbor for hyperspectral image classification. *Geoscience and Remote Sensing, IEEE Transactions on* 48(11):4099–4109.

Mensink, T.; Verbeek, J.; Perronnin, F.; and Csurka, G. 2012. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Computer Vision–ECCV 2012*. Springer. 488–501.

Nene, Sameer, A.; Nayar, Shree, K.; and Hiroshi, M. 1996. Columbia object image library (coil-20). Technical report, technical report CUCS-005-96.

Parameswaran, S., and Weinberger, Kilian, Q. 2010. Large margin multi-task metric learning. In *Advances in neural information processing systems*, 1867–1875.

Perrot, M., and A., H. 2015. Regressive virtual metric learning. In *Advances in Neural Information Processing Systems*. 1810–1818.

Torresani, L., and Lee, K.-c. 2006. Large margin component analysis. In *Advances in neural information processing systems*, 1385–1392.

Vandenberghe, L., and Boyd, S. 1996. Semidefinite programming. *SIAM review* 38(1):49–95.

Wang, H.; Nie, F.; and Huang, H. 2014. Robust distance metric learning via simultaneous l1-norm minimization and maximization. In *International Conference on Machine Learning*, 1836–1844.

Weinberger, Kilian, Q., and Saul, Lawrence, K. 2009. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research* 10:207–244.

Weinberger, Kilian, Q.; Blitzer, J.; and Saul, Lawrence, K. 2005. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, 1473–1480.

Xiang, S.; Nie, F.; and Zhang, C. 2008. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition* 41(12):3600–3612.

Xing; Eric, P.; Andrew, Y.; Jordan, Michael, I.; and Stuart, R. 2003. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems* 15:505–512.

Yang, L.; Jin, R., and Sukthankar, R. 2006. An efficient algorithm for local distance metric learning. *AAAI* 543–548.

You, J.; Wu, A.; Li, X.; and Zheng, W.-s. 2016. Top-push video-based person re-identification. 1345–1353.

Yuan, S.; Bellet, A.; and Fei, S. 2014. Sparse compositional metric learning. *arXiv preprint* 1404.4105.