Mixtures of Large Margin Nearest Neighbor Classifiers

Murat Semerci and Ethem Alpaydın

Department of Computer Engineering
Boğaziçi University
TR-34342 Istanbul, Turkey
{murat.semerci,alpaydin}@boun.edu.tr

Abstract. The accuracy of the k-nearest neighbor algorithm depends on the distance function used to measure similarity between instances. Methods have been proposed in the literature to learn a good distance function from a labelled training set. One such method is the large margin nearest neighbor classifier that learns a global Mahalanobis distance. We propose a mixture of such classifiers where a gating function divides the input space into regions and a separate distance function is learned in each region in a lower dimensional manifold. We show that such an extension improves accuracy and allows visualization.

Keywords: Nearest Neighbor Classifier, Margin Loss, Distance Learning

1 Introduction

Nonparametric, memory-based methods, such as the k-nearest neighbor classifier, interpolates from past similar cases. This requires a good distance (or inversely, similarity) measure to determine the relevant subset of the training set. Given two d-dimensional instances $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Re^d$, the Euclidean distance, or its square, is the best known:

$$D_E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)$$

The Euclidean distance assumes that all features have the same variance and that they are uncorrelated. If this is not the case and there is a covariance structure as given by a covariance matrix $\mathbf{S}$, one should use the Mahalanobis distance:

$$D_{\mathbf{M}} = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)$$

where $\mathbf{M} \equiv \mathbf{S}^{-1}$. The Euclidean distance is a special case where $\mathbf{M} = \mathbf{S} = \mathbf{I}$, the identity matrix.

$\mathbf{M}$ is a $d \times d$ symmetric, positive semi-definite matrix and when d is large, not all features may be informative and/or there may be strong correlations

between features, and one may want to do dimensionality reduction by a low-rank approximation. Any symmetric Mahalanobis matrix can be factorized as $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, where $\mathbf{L}$ is an $e \times d$ projection matrix and $e \leq d$:

$$\begin{aligned} D_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{M} (\boldsymbol{x}_i - \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{L}^\top \mathbf{L} (\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= (\mathbf{L}(\boldsymbol{x}_i - \boldsymbol{x}_j))^\top \mathbf{L}(\boldsymbol{x}_i - \boldsymbol{x}_j) = (\mathbf{L}\boldsymbol{x}_i - \mathbf{L}\boldsymbol{x}_j)^\top (\mathbf{L}\boldsymbol{x}_i - \mathbf{L}\boldsymbol{x}_j) \\ &= \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 = D_E(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ &= D_{\mathbf{L}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{1}$$

That is, using such a low-rank ($e < d$) approximation is equivalent to projecting the data to this new e-dimensional space as, $\boldsymbol{z}_i = \mathbf{L}\boldsymbol{x}_i$, $\boldsymbol{z}_i \in \Re^e$ and using Euclidean distance there.

In a high dimensional problem, different regions of the input space may exhibit different dependencies and variances and hence, instead of a single, global metric, it may be more appropriate to use different metrics in different regions. Besides, because regions have local structures, dimensionality can be further decreased. In this study, we propose a framework where the input space is partitioned into regions and different projection matrices are learnt in different regions.

The rest of this paper is organized as follows: We give a brief literature survey of related work in Section 2, and among these, the closest to our work are the Large Margin Nearest Neighbor (LMNN) algorithm—that learns $\mathbf{M}$—and Large Margin Component Analysis (LMCA) algorithm—that learns $\mathbf{L}$—which are discussed in more detail Section 3. Our proposed extension of mixtures of LMNN—that learns multiple $\mathbf{M}_m$ or $\mathbf{L}_m$ in different parts of the input space—is given in Section 4. We discuss our experimental results in Section 5 and conclude in Section 6.

2 Related Work

In the literature, many methods have been proposed to train a Mahalanobis matrix $\mathbf{M}$ or a projection matrix $\mathbf{L}$. Some methods train multiple Mahalanobis or projection matrices, which can be per-class or per-exemplar. Below, chronologically we give a brief summary of some methods.

One of the first distance metric learning algorithm is given by Xing et al. in [1] who define a convex optimization problem to find a Mahalanobis matrix. The instances in the data set form two disjoint subsets of similar and dissimilar pairs and a Mahalanobis matrix is trained such that the distance between similar points is minimized while the dissimilar points are at least 1 unit way from each other.

Neighborhood Components Analysis (NCA) is a stochastic gradient-based algorithm to find a linear projection matrix that minimizes the leave-one-out classification error of the nearest neighbor classifier in the new space (see [2]). A differentiable objective function is defined on the soft neighbor assignments in

the linearly projected space. The projection matrix can be also used for dimensionality reduction. Slakhutdinov and Hinton extend NCA to embed a nonlinear projection in [3], where a multilayer neural network is trained for this purpose.

Frome et al. in [4] propose a method to train a weight vector for each image to calculate the global distance between images using the feature vector which is a concatenation of local patch distances between the images. A large margin classifier is trained over the local distance feature vectors in a convex programming problem. Since the trained distances are not guaranteed to be compatible, a logistic model is trained over them to estimate posterior probabilities. They are ranked and the query image is assigned to the class of the image with the highest rank. In [5], they improve the algorithm by training a globally consistent local distance functions such that no second-level classifier is required to be trained. They redefine the problem using a convex optimization formulation. Since all the weight vectors are trained together, the final distance estimates are compatible with each other.

Chang and Yeung in [6] train an affine function per instance that provides smooth transitions between instances. A variant of regularized moving least squares is applied in a semi-supervised setting. Although the objective function has a closed-form solution, it becomes intractable for data sets with many instances and an approximation algorithm is given.

Davis et al. in [7] study metric learning from an information-theoretic point of view. They define the optimal Gaussian distribution whose covariance matrix satisfies the distance constraints defined on the instance pairs; the distance between instance pairs belonging to the same class must be smaller than a predetermined threshold and the instance pairs from different classes must be away from each other by at least a specified distance. Then, the problem is converted into a LogDet (logarithm of determinant) optimization problem that is convex.

The Large Margin Nearest Neighbor algorithm (LMNN) (see [8] and [9]) defines a semi-definite programming problem over the squared Mahalanobis distances of target and impostor sets—the impostors are the closest instances with different class labels and targets are the closest instances with the same label. Distances to the target neighbors are minimized while the distances to impostors are penalized if they are within a margin, which is a safe distance away from the furthest target neighbor. This is a convex programming instance and hence has a unique solution. A multiple metrics version of LMNN where a metric is trained for each class is also studied in [9].

Large Margin Component Analysis (LMCA) in [10] is a variant of LMNN and finds a lower dimensional rectangular projection matrix $\mathbf{L}$ instead of a square Mahalanobis matrix $\mathbf{M}$ (see Eq.1). Both methods share the same objective function but since LMCA defines the squared distance in terms of the projection matrix, this is no longer a convex optimization problem and LMCA converges to a local optimum. Our proposed method is an extension of LMNN and LMCA, and these methods will be discussed in more detail in Section 3, before we discuss our method in Section 4.

Malisiewicz and Efros in [11] focus on training per-exemplar metric for image retrieval. They also work on the concatenated vector of segment distances. Their algorithm consists of two parts. Sequentially, they train metrics per-exemplar given the nearest neighbors and then they re-assign the nearest neighbors given the trained metrics. They specify a margin on the trained distance function values as used in support vector machines (SVM). The neighbors that are away less than one unit distance are called the support set and the precision of classification result is determined by this support set.

Zhan et al. in [12] propose to learn instance-specific distances by using metric propagation. A smooth function (such as a Gaussian kernel) is propagated between the labelled and unlabelled instances. A regularized loss function is defined such that the distances between instances of the same label are minimized with respect to the given neighborhood relationships; the distance function trained using labeled instances can then be used for unlabeled instances. The proposed framework is formulated as a convex problem.

Chen and Sun in [13] propose a hierarchical LMNN algorithm. Overlapping ratio is defined to measure the confusion between classes and if this ratio is above a threshold, overlapping classes are grouped in the same cluster. The hierarchy describes how to map a test instance to a cluster. A Mahalanobis matrix is trained for each cluster and a given test instance is classified by using its cluster's metric matrix.

Noh et al. in [14], aim reducing the expectation error that the nearest neighbor has a different label. They show that if the distributions of the two classes are known, the difference between the empirical nearest neighbor error and the optimal nearest neighbor error based on asymptotic Bayes error is caused by finite sampling. They propose Generative Local Metric Learning which defines a convex problem if the divergence function used is also convex.

Chang in [15] proposes an iterative metric boosting method. An upper bound function on the leave-one-out error for the nearest neighbor classification is defined and is minimized. The misclassified instances are weighted and the Mahalanobis matrix is optimized with respect to these weights. An eigenvalue problem is solved to find the Mahalanobis matrix.

Bunte et al. in [16] propose Limited Rank Matrix Learning which is a recent algorithm that extends Learning Vector Quantization. It learns class prototypes and a low-rank projection matrix at the same time, iteratively. The projection matrix is trained to be discriminative by optimizing a cost function that maps instances close to their class prototypes and away from the other class prototypes, using a criterion similar to that of Linear Discriminant Analysis.

Wang et al., in [17], propose to combine multiple metric matrices to form per-exemplar local metrics. The algorithm consists of two steps. First, a weight matrix is trained such that each data point can be expressed as a linear combination of pre-defined anchor points. The cluster means are defined as the anchors and any clustering algorithm can be used for defining the anchors, i.e., k-means. Then, a metric learning algorithm, a modified version of Multiple-metric LMNN, is used to train a metric for each anchor point. The per-exemplar local metrics

are combinations of these anchor metrics whose weights are determined by the weight matrix.

The Bregman distance functions are trained in a SVM-like manner in [18]. Since the general Bregman distances are not metrics, the authors work with a particular set of convex Bregman functions to ensure that they train a metric. Kernelizing the Bregman distances, they solve a quadratic problem.

Table 1. The overall summary of distance metric learning methods

Method	Convexity	Type of Metric	Distance
Xing et al. [1]	Yes	Single	Mahalanobis
Golberger et al. [2]	No	Single	Projection
Salakhutdinov and Hinton [3]	No	Single	Nonlinear Projection
Frome et al. [4]	Partial	Per-Exemplar	Weight Vector
Frome et al. [5]	Yes	Per-Exemplar	Weight Vector
Chang and Yeung [6]	Yes	Per-Exemplar	Mahalanobis
Davis et al. [7]	Yes	Single	Mahalanobis
Weinberger and Saul [8]	Yes	Single	Mahalanobis
Weinberger and Saul [9]	Yes	Per-Class	Mahalanobis
Torresani and Lee [10]	No	Single	Projection
Malisiewicz and Efros [11]	Partial	Per-Exemplar	Weight Vector
Zhan et al. [12]	Yes	Per-Exemplar	Weight Vector
Chen and Sun [13]	Partial	Per-Cluster	Mahalanobis
Noh et al. [14]	Yes	Single	Mahalanobis
Chang [15]	Partial	Single	Mahalanobis
Bunte et al. [16]	No	Per-Class Single	Vector Projection
Wang et al. [17]	Partial	Multiple	Mahalanobis
Wu et al. [18]	Yes	Per-Exemplar	Mahalanobis

The methods are summarized in Table 1. Note that partial convexity means that the algorithm consists of some sub-problems or steps and that not all of them are convex.

3 Large Margin Nearest Neighbor (LMNN) and Large Margin Component Analysis (LMCA) Algorithms

The Large Margin Nearest Neighbor (LMNN) trains a global Mahalanobis matrix that evaluates distances discriminatively (see [9] and [8]). Let us define our data set as pairs $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is the input instance vector and y_i is the corresponding label. The notation $j \rightsquigarrow i$ (j leads to i) means $\boldsymbol{x}_j$ is a target neighbor of $\boldsymbol{x}_i$. A *target* is a neighbor with the same (correct) class label whereas an *impostor* is a neighbor with different (wrong) class label. For accurate nearest neighbor classification, targets must be closer than the impostors.

Using the label information, a Mahalanobis matrix $\mathbf{M}$ can be trained to

$$\begin{aligned} \text{minimize} \quad & (1-\mu)\sum_{i,j\rightsquigarrow i}(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_j)+\mu\sum_{i,j\rightsquigarrow i,l}(1-y_{il})\xi_{ijl} \\ \text{subject to} \quad & (\boldsymbol{x}_i-\boldsymbol{x}_l)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_l)-(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_j)\geq 1-\xi_{ijl} \\ & \xi_{ijl}\geq 0 \\ & \mathbf{M}\succeq 0 \end{aligned} \tag{2}$$

where $y_{il}=1$ if $y_i=y_l$, which are the labels of $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$, and $y_{il}=0$ otherwise. The first term is the sum of distances of each instance to its target neighbors which we want to be minimum and the second term penalizes close impostors: For any instance i where l is an impostor and j is a target, we would like the distance to the impostor be at least one unit more than the distance to a target. If this is not satisfied, there is a slack and we minimize the sum of such slacks.

Equation 2 defines a positive semi-definite programming problem and there is a unique minimum. After some manipulations, the loss function can be rewritten as:

$$\begin{aligned} E=(1-\mu)\sum_{i,j\rightsquigarrow i}&\text{trace}(\mathbf{M}\mathbf{C}_{ij}) \\ &+\mu\sum_{i,j\rightsquigarrow i,l}(1-y_{il})\left[1+\text{trace}(\mathbf{M}\mathbf{C}_{ij})-\text{trace}(\mathbf{M}\mathbf{C}_{il})\right]_+ \end{aligned} \tag{3}$$

where $[a]_+$ is the hinge loss which is a when $a>0$ and is 0 otherwise. The difference matrix, $\mathbf{C}_{ij}$, is defined as $\mathbf{C}_{ij}=(\boldsymbol{x}_i-\boldsymbol{x}_j)(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top$. Though other solving methods such as alternating projection algorithms can also be used here, using iterative gradient descent is simple and the global solution can still be reached [8]. The gradient is:

$$\frac{\partial E}{\partial \mathbf{M}}=(1-\mu)\sum_{i,j\rightsquigarrow i}\mathbf{C}_{ij}+\mu\sum_{(i,j,l)}(\mathbf{C}_{ij}-\mathbf{C}_{il}) \tag{4}$$

where (i,j,l) means active triples (that activate the hinge loss) in the current gradient update (the impostors can vary in each update).

As we discussed in Equation 1, the metric matrix learned can be factorized as $\mathbf{M}=\mathbf{L}^\top\mathbf{L}$, where $\mathbf{L}$ is the projection matrix. Large margin component analysis (LMCA) in [10] is a variant of LMNN which uses this idea. It focuses on finding a lower dimensional rectangular projection matrix instead of a full square Mahalanobis matrix. LMCA also minimizes Equation 3, but when defined in terms of $\mathbf{L}$, this is no longer a convex optimization problem and gradient-descent is used. At each iteration, the projection matrix is updated in the negative direction of the gradient:

$$\frac{\partial E}{\partial \mathbf{L}}=2(1-\mu)\mathbf{L}\sum_{i,j\rightsquigarrow i}\mathbf{C}_{ij}+2\mu\mathbf{L}\sum_{(i,j,l)}(\mathbf{C}_{ij}-\mathbf{C}_{il}) \tag{5}$$

4 Mixtures of Large Margin Nearest Neighbor Classifiers

LMNN uses the a single, global $\mathbf{M}$ and LMCA uses a single, global $\mathbf{L}$ over the whole input space. It may be the case that a data set has multiple locally varying distributions—features may have different variances and different correlations in different parts of the input space, defining multiple local manifolds. Our idea is to divide up the input space into local regions using a gating function and learn different metrics in different regions; we hence define a mixture of LMNNs. In doing this, we are inspired by the Mixture of Experts neural network model of Jacobs et al. in [19]. Previously, Gönen and Alpaydın in [20] used the same idea in multiple kernel learning where they write a kernel as a weighted sum of localized kernels.

The gating function that defines the region of expertise of a local metric can be *cooperative* or *competitive*, which is implemented respectively by the sigmoid or softmax function (P is the number of regions):

$$\text{Cooperative: } \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m) = \frac{1}{1+\exp(-\boldsymbol{w}_m^\top \boldsymbol{x}_i - w_{m0})} \tag{6}$$

$$\text{Competitive: } \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m) = \frac{\exp(\boldsymbol{w}_m^\top \boldsymbol{x}_i + w_{m0})}{\sum_{h=1}^{P}\exp(\boldsymbol{w}_h^\top \boldsymbol{x}_i + w_{h0})} \tag{7}$$

Local model m becomes active if $\eta_m(\boldsymbol{x}_i) > 0$ and we say that $\boldsymbol{x}_i$ belongs to region m. The softmax function is competitive because it enforces a soft winner-take-all mechanism and for any input, we expect a single active local metric and the gating model works as a selector ($\sum_m \eta_m(\boldsymbol{x}_i) = 1$). The sigmoid function is cooperative because there can be more than one active local metric and the model takes a weighted sum ($\sum_m \eta_m(\boldsymbol{x}_i)$ need not be 1).

In each local region, using a full $\mathbf{M}$ may lead to overfitting and to regularize, we learn a local lower rank $\mathbf{L}$ in each: When $\boldsymbol{x}$ chooses local model m, $\mathbf{L}_m$ is the local projection used. The localized projection of $\boldsymbol{x}_i$ into region m is

$$\boldsymbol{z}_{i,m} = \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\mathbf{L}_m\boldsymbol{x}_i$$

The *total distance* between a pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is calculated as the sum of the local distances:

$$D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} D_{\mathbf{L}_m}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

where $D_{\mathbf{L}_m}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the local distance in region m:

$$\begin{aligned} D_{\mathbf{L}_m}(\boldsymbol{x}_i, \boldsymbol{x}_j) =& \|\boldsymbol{z}_{i,m} - \boldsymbol{z}_{j,m}\|_2^2 \\ =& \|\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\mathbf{L}_m\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\mathbf{L}_m\boldsymbol{x}_j\|_2^2 \\ =& \|\mathbf{L}_m(\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j)\|_2^2 \\ =& \left[\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j\right]^\top \mathbf{L}_m^\top \\ & \mathbf{L}_m\left[\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j\right] \end{aligned} \tag{8}$$

Hence, the total distance is a weighted combination of local distances and the contribution of local projections are determined by $\eta_m(\boldsymbol{x}_i)$. Thus, it is possible that multiple metrics are active, particularly in the cooperative setting.

The model parameters are the localized projection matrices $\mathbf{L}_m$ and the gating parameters $\boldsymbol{w}_m$. We use the same formulation of LMNN in Equation 2 by using D_{total} instead of the Mahalanobis distance $(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{M}(\boldsymbol{x}_i - \boldsymbol{x}_j)$:

$$\begin{aligned} \text{minimize} \quad & (1-\mu) \sum_{i,j \rightsquigarrow i} D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \mu \sum_{i,j \rightsquigarrow i,l} (1 - y_{il}) \xi_{ijl} \\ \text{subject to} \quad & D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_l) - D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \end{aligned} \tag{9}$$

Let us rewrite the loss function:

$$E(\eta) = (1-\mu) \sum_{i,j \rightsquigarrow i} \sum_{m=1}^{P} \text{trace}(\mathbf{L}_m^\top \mathbf{L}_m \mathbf{C}_{ij}^{(m)}(\eta)) + \mu \sum_{i,j \rightsquigarrow i,l} (1 - y_{il}) \left[1 + \zeta_{ijl}\right]_+ \tag{10}$$

where

$$\zeta_{ijl} = \sum_{m=1}^{P} \left(\text{trace}(\mathbf{L}_m^\top \mathbf{L}_m \mathbf{C}_{ij}^{(m)}(\eta)) - \text{trace}(\mathbf{L}_m^\top \mathbf{L}_m \mathbf{C}_{il}^{(m)}(\eta)) \right)$$

We can use the same trick and rewrite the gated loss function in terms of difference matrices. $\mathbf{C}_{ij}^{(m)}(\eta)$ is defined over the gated projections of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in region m:

$$\mathbf{C}_{ij}^{(m)}(\eta) = \left[\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j\right] \left[\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j\right]^\top$$

When we use a gating function, the problem is not convex anymore and we use gradient descent. The derivative of the loss function with respect to the local projection matrix $\mathbf{L}_m$ can then be derived:

$$\frac{\partial E(\eta)}{\partial \mathbf{L}_m} = 2(1-\mu)\mathbf{L}_m \sum_{i,j \rightsquigarrow i} \mathbf{C}_{ij}^{(m)}(\eta) + 2\mu \mathbf{L}_m \sum_{(i,j,l)} (\mathbf{C}_{ij}^{(m)}(\eta) - \mathbf{C}_{il}^{(m)}(\eta)) \tag{11}$$

The derivative of objective function with respect to the gating parameters depends on the function used:

$$\begin{aligned} \frac{\partial E(\eta)}{\partial w_{h,k}} = & 2(1-\mu) \sum_{i,j \rightsquigarrow i} \frac{\partial D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial w_{h,k}} \\ & + \mu \sum_{(i,j,l)} (1 - y_{il}) \left(\frac{\partial D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial w_{h,k}} - \frac{\partial D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_l)}{\partial w_{h,k}} \right) \end{aligned} \tag{12}$$

Algorithm 1 Training a Mixture of Large Margin Nearest Neighbor Classifiers

```
Initialize w_{m,k} and w_{m,0} to small random numbers.
Initialize L_m matrices to the PCA projection matrix of the whole data.
repeat
    repeat
        Calculate D(x_i, x_j) and find target neighbors and impostors.
        w_{m,k}^{(t+1)} ← w_{m,k}^{(t)} − γ^{(t)} ∂E(η)/∂w_{m,k}
    until convergence of gating parameters
    repeat
        Calculate D(x_i, x_j) and find target neighbors and impostors.
        L_m^{(t+1)} ← L_m^{(t)} − γ^{(t)} ∂E(η)/∂L_m
    until convergence of local projections
until convergence
```

We can apply the chain rule to get the derivative of the total distance:

$$\frac{\partial D_{total}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial w_{h,k}} = \sum_{m=1}^{P} 2 \left[\boldsymbol{x}_i \frac{\partial \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)}{\partial w_{h,k}} - \boldsymbol{x}_j \frac{\partial \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)}{\partial w_{h,k}} \right]^\top \mathbf{L}_m^\top \mathbf{L}_m \left[\eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\boldsymbol{x}_i - \eta_m(\boldsymbol{x}_j|\boldsymbol{w}_m)\boldsymbol{x}_j \right] \tag{13}$$

If the sigmoid gating is used, the derivatives are ($x_{i,0} \equiv 1$):

$$\frac{\partial \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)}{\partial w_{h,k}} = \delta_{mh} \left(1 - \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)\right) \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m) x_{i,k}, \;\; k = 0, 1, \ldots, d \tag{14}$$

For the softmax gating, we have ($x_{i,0} \equiv 1$):

$$\frac{\partial \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m)}{\partial w_{h,k}} = \left(\delta_{mh} - \eta_h(\boldsymbol{x}_i|\boldsymbol{w}_h)\right) \eta_m(\boldsymbol{x}_i|\boldsymbol{w}_m) x_{i,k}, \;\; k = 0, 1, \ldots, d \tag{15}$$

where δ_{mh}, is 1 if $m = h$ and it is 0 otherwise.

The pseudo-code for the Mixture of LMNN (MoLMNN) is given in Algorithm 1. To have a meaningful starting projection direction we initialize the local projections $\mathbf{L}_m$ by using Principal Components Analysis (PCA) on the training data. At each iteration, we first apply gradient-descent to update the gating parameters, and then, using the trained gating model, the local projection matrices are updated. We apply these steps, until both the gating model and the projection matrices converge or the classification result does not improve any further. The learning rate, γ, is determined using linear search at each iteration.

5 Experiments and Results

We compare sigmoid and softmax-gated MoLMNN with LMNN and LMCA on 21 data sets, that are publicly available in [20, 21, 22, 23]. In yeast, faults and segment data sets, two classes are used (nuc vs cyt, k_stratch vs bumps, and

sky vs windows, respectively). In musk data set, only the real valued features are used. The input data is z-normalized.

Our experimental methodology is as follows: Each data set is split into two subsets as one-third test data and two-thirds training and validation data. The two-thirds part is used to create ten training and validation folds using 5×2 cross-validation. The number of reduced dimensions, namely e, is chosen among the number of features that explain 90, 95 and 98 per cent of the variance; LMNN, LMCA and MoLMNNs models with $P = 1$ up to 10 regions are trained for $k = 3, 5, 7$ and 9 neighbors . We also try and choose the best of sigmoid and softmax gating. We do such a four-dimensional, $(k, e, P, \text{sigmoid/softmax})$ grid search and choose the combination that has the highest average validation accuracy—the other models are similarly trained and the best configuration is chosen for their parameter set. For the best setting, the corresponding model is trained on the ten different training folds and tested on the same left-out one-third test set. These ten test results are reported and compared with the parametric 5×2 cross-validation paired F test [24]. Table 2 shows the mean and standard deviation of the test results for each data set and the results of significance tests.

We see that on most data sets, a few regions ($P \leq 4$) is enough. The number of regions correspond to the modalities of data with different input distributions. Increasing the number of regions does not improve accuracy beyond a certain value. Note that even with a single region, MoLMNN may be more accurate because it reassigns impostors and targets at each iteration while the other algorithms fix them at initialization.

We also find that MoLMNN uses more neighbors when compared with other algorithms. We believe this to be an indicator that MoLMNN trains a more suitable distance function which places more of the target neighbors nearby. Other algorithms use fewer nearest neighbors because due to inaccurate distance approximation, their performance degrade if more neighbors are used. In terms of sigmoid vs softmax gating, we do not notice one being always superior to the other—each has its use.

MoLMNN significantly outperforms both LMCA and LMNN on Arabidopsis, Musk, Yeast and Sonar. It outperforms LMCA on Splice and LMNN on Yale. LMCA gives higher accuracy results on Yale and Ionosphere data sets. Except Yeast, these data sets have more than 60 dimensions, which shows that MoLMNN can capture local information to improve performance.

On the arabidopsis data set, we can visualize the data by reducing dimensionality to two; this is a bioinformatics data set with $1,000$ dimensions. In Fig. 1(a), we see the plot using PCA; there we see that the data has significant structure but that PCA cannot capture the difference between the two classes. In Fig. 1(b), we see the plot using LMCA and the learned discriminant using $k = 3$. In Fig. 2, we see results with MoLMNN with two regions. Each data point is plotted in the region where its gating value is higher and the discriminants are plotted separately in each region with $k = 3$. We see that we get better discrimination between the classes this way. Though we have not checked for this application,

Table 2. The mean and standard deviation of test set accuracies of MoLMNN, LMCA and LMNN. The parameters of the best configuration are given in parantheses, where So is softmax and Si is sigmoid. Boldface indicates that the method is significantly better than the other two. In terms of pairwise comparisons (shown by '*'), on splice, MoLMNN is more accurate than LMCA and on Ionosophere, LMCA is more accurate than MoLMNN.

Data set	**MoLMNN (k,e,P,g)**	**LMCA (k,e)**	**LMNN (k,d)**
Abalone	77.86 ± 1.01 (9,3,1,So)	78.21 ± 1.97 (9,3)	78.03 ± 1.33 (9,7)
Arabidopsis	**81.89 ± 0.97** (9,473,1,Si)	77.21 ± 1.92 (5,390)	69.24 ± 2.17 (3,1000)
Australian	86.17 ± 1.42 (9,14,7,So)	85.83 ± 2.15 (5,14)	86.43 ± 1.08 (9,14)
Bupa	61.65 ± 4.27 (5,6,4,Si)	58.61 ± 2.79 (9,5)	59.57 ± 3.13 (9,6)
Ctg	89.73 ± 0.80 (9,16,2,So)	89.96 ± 0.62 (5,11)	89.83 ± 0.71 (5,21)
Faults	98.52 ± 0.58 (9,14,4,Si)	98.48 ± 0.25 (9,14)	98.56 ± 0.16 (9,27)
German Numeric	70.21 ± 2.27 (9,20,1,Si)	72.10 ± 2.84 (9,18)	72.01 ± 1.95 (9,24)
Heart	83.89 ± 4.10 (5,3,5,Si)	82.44 ± 4.38 (5,4)	85.22 ± 3.10 (7,13)
Ionosphere	81.54 ± 2.36 (9,17,3,Si)	83.25 ± 2.65* (3,27)	82.91 ± 2.88 (3,34)
Mg	83.58 ± 0.79 (9,5,1,So)	82.34 ± 0.61 (3,6)	82.34 ± 0.61 (3,6)
Musk	**86.01 ± 2.66** (7,17,4,So)	80.82 ± 2.88 (5,28)	79.62 ± 4.00 (5,166)
Optdigits	96.98 ± 0.31 (9,51,3,Si)	97.26 ± 0.35 (3,41)	97.33 ± 0.35 (3,64)
Pendigits	97.33 ± 0.31 (5,11,2,So)	97.49 ± 0.15 (3,11)	97.31 ± 0.23 (5,16)
Pima	72.93 ± 0.96 (9,8,2,So)	73.32 ± 1.90 (9,8)	73.32 ± 1.90 (9,8)
Segment	99.95 ± 0.14 (7,8,9,Si)	100.00 ± 0.00 (7,8)	100.00 ± 0.00 (7,19)
Sonar	**76.52 ± 3.60** (7,36,10,Si)	71.16 ± 3.01 (3,28)	68.55 ± 5.11 (3,60)
Splice	89.03 ± 0.71* (9,50,1,Si)	84.97 ± 0.83 (5,58)	85.81 ± 0.76 (9,60)
Transfusion	78.67 ± 1.13 (9,3,2,Si)	79.40 ± 1.34 (9,3)	79.36 ± 1.37 (9,5)
Wdbc	94.97 ± 1.35 (9,14,4,So)	94.02 ± 2.23 (7,10)	94.29 ± 1.94 (3,30)
Yeast	**60.57 ± 2.58** (9,6,6,Si)	59.39 ± 2.38 (5,6)	59.33 ± 2.52 (5,8)
Yale	93.51 ± 0.77 (3,196,3,Si)	**94.90 ± 0.50** (3,88)	92.77 ± 0.56 (7,896)

where the gating boundary lies and the dimensions in each region could also carry information.

We also check the relationship between the number of regions and the number of reduced dimensions. Figure 3 shows how test accuracy changes as we vary the number of dimensions and the number of regions. This is for $k = 9$, but we see similar behavior for other k. We see that it is more the number of regions that affect accuracy rather than the local dimensionality; we also see that sigmoid gating leads to more fluctuating performance—regions may overlap and hence may interfere more.

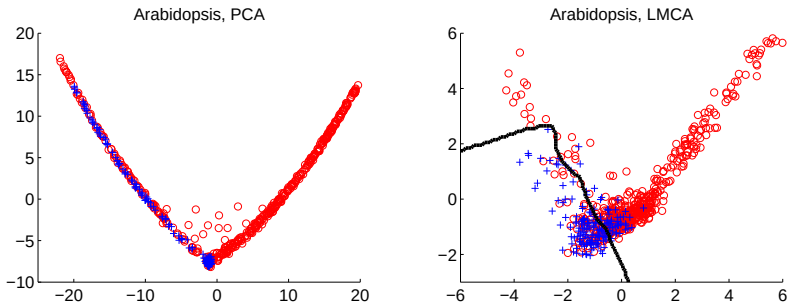
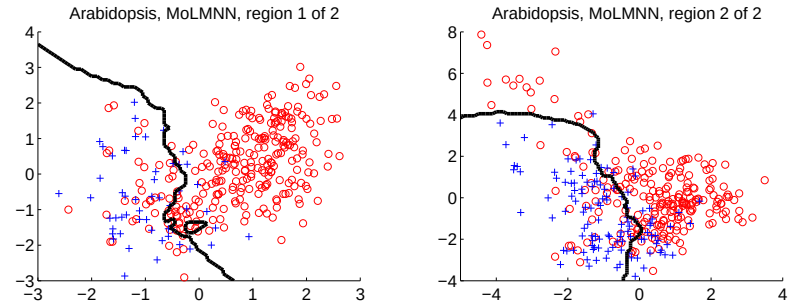
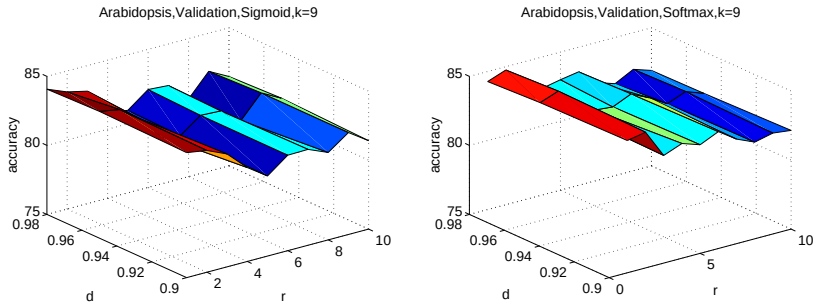


Fig. 1. The 2d mappings of Arabidopsis data set with (a) PCA and (b) LMCA.



(a) MoLMNN 1^{st} region (b) MoLMNN 2^{nd} region

Fig. 2. The 2d mappings of Arabidopsis data set with MoLMNN (softmax) with two regions.



(a) Sigmoid (b) Softmax

Fig. 3. The effects of the number of regions and proportion of variance explained on accuracy on validation dataset when $k = 9$.

6 Conclusions

In this study, we propose the Mixture of LMNN (MoLMNN) method which softly partitions the input space and trains a separate projection matrix in each region to best discriminate the data. The partitioning of the space and the training of the projection matrices are coupled. Our experiments on real data sets show that compared with LMNN and LMCA proper, the mixture approach frequently performs better. Localization of the data and reducing dimensionality to two allows visualization. The boundary of the gating model and the projected dimensions could carry information which may help understand the data.

Acknowledgements This work is supported by TÜBITAK Project EEEAG 109E186 and Boğaziçi University Research Funds BAP5701.

References

[1] Xing, P.E., Ng, A.Y., Jordan, M.I., Russell, S.: Distance Metric Learning, with Application to Clustering with Side-Information. In : Advances in Neural Information Processing Systems 15, Cambridge, MA, MIT Press. 505–512 (2002)

[2] Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood Components Analysis. In: Advances in Neural Information Processing Systems 17, MIT Press 513–520 (2004)

[3] Salakhutdinov, R., Hinton, G.: Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In: 11th International Conference on Artificial Intelligence and Statistics, Volume 2. 412–419 (2007)

[4] Frome, A., Singer, Y., Malik, J.: Image Retrieval and Classification Using Local Distance Functions. In Schölkopf, B., Platt, J., Hoffman, T., eds.: Advances in Neural Information Processing Systems 19, Cambridge, MA, MIT Press 417–424 (2007)

[5] Frome, A., Singer, Y., Sha, F., Malik, J.: Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. In: 11st IEEE International Conference on Computer Vision. 1–8 (2007)

[6] Chang, H., Yeung, D.Y.: Locally Smooth Metric Learning with Application to Image Retrieval. In: 11th IEEE International Conference on Computer Vision. 1–7 (2007)
[7] Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-Theoretic Metric Learning. In: 24th International Conference on Machine Learning. 209–216 (2007)
[8] Weinberger, K.Q., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. Journal of Machine Learning Research **10**, 207–244 (2009)
[9] Weinberger, K.Q., Saul, L.K.: Fast Solvers and Efficient Implementations for Distance Metric Learning. In: 25th International Conference on Machine Learning. 1160–1167 (2008)
[10] Torresani, L., Lee, K.C.: Large Margin Component Analysis. In: Advances in Neural Information Processing Systems 19. 1385–1392 (2007)
[11] Malisiewicz, T., Efros, A.A.: Recognition by Association via Learning Per-Exemplar Distances. In: 21st IEEE Conference on Computer Vision and Pattern Recognition. 1–8 (2008)
[12] Zhan, D.C., Li, M., Li, Y.F., Zhou, Z.H.: Learning Instance Specific Distances Using Metric Propagation. In: 26th International Conference on Machine Learning. 1225–1232 (2009)
[13] Chen, Q., Sun, S.: Hierarchical Large Margin Nearest Neighbor Classification. In: the 20th International Conference on Pattern Recognition. 906–909 (2010)
[14] Noh, Y.K., Zhang, B.T., Lee, D.D.: Generative Local Metric Learning for Nearest Neighbor Classification. In: Advances in Neural Information Processing Systems 23, MIT Press. 1822–1830 (2010)
[15] Chang, C.C.: A Boosting Approach for Supervised Mahalanobis Distance Metric Learning. Pattern Recognition **45**(2), 844–862 (2012)
[16] Bunte, K., Schneider, P., Hammer, B., Schleif, F.M., Villmann, T., Biehl, M.: Limited Rank Matrix Learning, Discriminative Dimension Reduction and Visualization. Neural Networks **26**, 159–173 (2012)
[17] Wang, Y., Woznica, A., Kalousis, A.: Parametric Local Metric Learning for Nearest Neighbor Classification. In: Advances in Neural Information Processing Systems 25, MIT Press. 1610–1618 (2012)
[18] Wu, L., Hoi, S.C.H., Jin, R., Zhu, J., Yu, N.: Learning Bregman Distance Functions for Semi-Supervised Clustering. IEEE Transactions on Knowledge and Data Engineering **24**, 478–491 (2012)
[19] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive Mixtures of Experts. Neural Computation **3**, 79–87 (1991)
[20] Gönen, M., Alpaydın, E.: Localized Multiple Kernel Learning. In: 25th International Conference on Machine Learning. 352–359 (2008)
[21] Frank, A., Asuncion, A.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/ (2010)
[22] Chang, C.C., Lin, C.J.: LIBSVM Data: Classification, Regression, and Multi-label. http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/ (2011)
[23] Sonnenburg, S., Ong, C.S., Henschel, S., Braun, M.: Machine Learning Data Set Repository. http://mldata.org/ (2011)
[24] Alpaydın, E.: Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms. Neural Computation **11**, 1885–1892 (1999)