Distance-based Mixture Modeling for Classification via Hypothetical Local Mapping

Mu Qiao* Jia Li†

Abstract

We propose a new approach for mixture modeling based only upon pairwise distances via the concept of hypothetical local mapping (HLM). This work is motivated by increasingly commonplace applications involving complex objects that cannot be effectively represented by tractable mathematical entities. The new modeling approach consists of two steps. A distance-based clustering algorithm is applied first. Then HLM takes as input the distances between the training data and their corresponding cluster centroids to estimate the model parameters. In the special case where all the training data are taken as cluster centroids, we obtain a distance-based counterpart of the kernel density. The classification performance of the mixture models is compared with other state-of-the-art distance-based classification methods. Results demonstrate that HLM-based algorithms are highly competitive in terms of classification accuracy and are computationally efficient. Furthermore, the HLM-based modeling approach adapts readily to incremental learning. We have developed and tested two schemes of incremental learning scalable for dynamic data arriving at a high speed.

Keywords: Distance-based; Mixture modeling; Classification; Hypothetical local mapping; Kernel density; Incremental learning; Complex data; Large-scale

1 Introduction

Distance-based classifiers categorize objects using only the pairwise distances between samples in the test and training data sets. Because only the pairwise distances are required in training and prediction, the representation of the instances can be of great variety or even unspecified. This feature is particularly appealing to classification problems where the object cannot be described effectively by a mathematical entity permitting well-studied analytical operations, for example, vectors. The term *distance* is used in a loose sense here for the pairwise similarity or dissimilarity relationship between data samples, not necessarily a true metric. Distance-based classification is widely used in the fields of computer vision [1], bioinformatics [2], information retrieval, etc.

*Mu Qiao is Research Staff Member at IBM Almaden Research Center, San Jose, CA 95120. Email: mqiao@us.ibm.com

†Jia Li is Professor in the Department of Statistics and (by courtesy) Computer Science and Engineering at The Pennsylvania State University, University Park, PA 16802. Email: jiali@stat.psu.edu

1.1 Overview of Our Work

In this paper, we investigate distance-based mixture modeling for classification using *hypothetical local mapping* (HLM), a mechanism originally proposed by Li and Wang [3] (refined in [4]). The major contribution of our current work is to extend Gaussian mixture modeling for the Euclidean space to a general distance-endowed space via HLM.

HLM has been successfully applied to model categories of images. The images are characterized by their color and texture signatures, each being a set of weighted and unordered vectors [4]. To estimate a mixture model, HLM takes as input the distances between the training image signatures and their corresponding cluster centroids. Conceptually, instances in one cluster and the cluster centroid are mapped to a Euclidean space, preserving maximally the pairwise distances. In the mapped space, this cluster of data is modeled by a Gaussian distribution with spherical covariance and a mean vector equal to the mapped centroid. The parameters of the Gaussian distribution and the dimension of the mapped space can be estimated by fitting a Gamma distribution using only distances between each data point and the cluster centroid in the original space. The distance-preserving mapping is thus bypassed, causing no additional computation, and hence called hypothetical. Finally, a mixture model is constructed to account for multiple clusters.

In the current work, we explore the potential of HLM as a mixture modeling technique in a more general setting than what has been pursued in [4]. In the previous work, the clustering algorithm exploits the mathematical representation of an image and is thus not pairwise distance-based. Although HLM only uses the distances after clustering, it has not been coupled with distance-based clustering algorithms. We hereby address this gap in the methodology of HLM and investigate its performance on more general data sets. Because the parameter fitting in HLM is computationally negligible in comparison with the pairwise distance-based clustering performed first, mixture modeling by HLM is almost as fast as the clustering process itself. With the existence of some fast distance-based clustering algorithms, HLM can be substantially faster than several major pairwise distance-based classification methods.

Moreover, noting the kinship between a finite mixture model and a kernel density, we find it natural to develop an HLM-based counterpart of the kernel density for a general space. In this case, every data point is treated as a centroid; and we propose a method to choose the bandwidth of the Gaussian kernel. This extension of the kernel density is especially useful for small training data prone to losing valuable information if clustering reduces the data even more. Finally, we push the consideration on computational efficiency one step further by proposing and evaluating two incremental learning schemes for HLM-based mixture models. The intrinsic characteristics of HLM lend it readily to incremental learning, an important scenario in light of the abundance of high velocity stream data.

1.2 Related Work

Our work is most related to distance-based generative models. Besides HLM exploited by Li and Wang [3, 4], another distance-based generative model was proposed by Cazzanti and Gupta [5], namely, local similarity discriminant analysis (local SDA). They assume that the expectation of the similarity between

a random sample X and a random class centroid μ equals the average similarity between the k nearest neighbors of X in the same class and the class centroid μ. Suppose the number of classes is K. The assumption of local SDA induces K^2 number of constraints. Given these constraints, each class-conditional distribution of X is estimated by the principle of maximum entropy. Cazzanti et al. [6] proposed a more generalized SDA by assuming that the expectation of the descriptive statistic of a random sample X with respect to the class-conditional distribution equals the average sample statistic over all the training data in each class. In SDA and local SDA, each class is modeled by a single parametric distribution. As an extension, mixture SDA was proposed [7], where K^2 number of mixture distributions have to be estimated. If the number of classes is large, the quadratic growth in the number of mixture distributions will hinder scalability. A multi-task regularized local SDA was later proposed [8, 9], which treats the estimation of different class-conditional distributions as multiple tasks and achieves the best performance among all the SDA-family algorithms.

As a generative mixture modeling approach, HLM is profoundly different from mixture SDA [7]. Because pairwise constraints on all the classes are assumed in mixture SDA, the distribution estimation for one class depends on all the other classes. Thus its complexity grows quickly with the number of classes. In contrast, HLM, along the line of mixture modeling, estimates the density of each class separately. In comparison with discriminative approaches, e.g., support vector machines (SVM) with pairwise distances modified into kernels [10, 11, 12], HLM inherits multiple advantages of the generative modeling approach, including the ease of handling a large number of classes, the convenience of incorporating domain expertise, and the minimal effort required to treat new classes in an incremental manner.

Other distance-based classification methods are not generative. K-nearest neighbor (k-NN) classifies a test sample based on its distances to all the training samples. The majority class of the k closest data points is assigned as the class label of that sample. K-NN is sensitive to noisy training samples especially when the distance measure is not well defined. To mitigate this issue, various distance measures or transformations have been proposed [13, 14, 15]. Several weighted versions of k-NN have also been proposed [12, 16, 17]. Another distance-based classification approach treats distances to training samples as features. Standard discriminative classifiers, such as SVM and Fisher linear discriminant analysis (LDA), are applied to these distance feature vectors [13, 18, 19]. A major limitation of this approach is that the dimension of the feature vector equals the number of training data, often prohibitively large. Additionally, as Chen et al. [12] found out, in the case of large inter-class variance relative to intra-class variance, treating distances as features may yield low discriminative power even when the classes are well separated.

If the pairwise distance matrix between training samples is symmetric and positive definite, it can be treated as a kernel and used in any kernel classifiers, e.g., SVM. Many distance matrices, however, do not satisfy these conditions. Several methods for modifying distances into kernels are discussed in [12]. Hochreiter and Obermayer [10] proposed potential support vector machines (P-SVM) applicable to any input data matrix, but the final kernels in these methods are $N \times N$ matrices, where N is the data size. As a result, P-SVM is computationally intensive with large N. A hybrid approach of SVM and k-NN [11] computes the pairwise distance matrix for the union of the test sample and its k nearest neighbors, modifies

the matrix into a kernel, and then applies standard SVM. One difficulty with modifying distances into kernels is that the original distances between the test and training samples have to be transformed so that they are consistent with the modified distances in producing the kernels.

The rest of the paper is organized as follows. Preliminaries on HLM and distance-based clustering methods are in Section 2. We propose two distance-based classification algorithms via HLM-based mixture modeling in Section 3. In Section 4, two HLM-based incremental learning schemes are presented. Section 5 is on experiments and results. Finally, we conclude in Section 6.

2 Preliminaries

We introduce in this section HLM-based mixture modeling for a general space and several distance-based clustering methods used in this work.

2.1 Hypothetical Local Mapping

Consider data points in a general space Ω provided with pairwise distances. To form a distribution for such data, there are two major schools of approaches in machine learning: discretization by clustering and mapping to the Euclidean space with the attempt of preserving distances. After discretization, a probability can be assigned to each cluster. The drawbacks of this approach include the neglect of difference between points in the same cluster and non-smooth transition from one cluster to another. These issues are addressed by the other approach of mapping. However, a Euclidean space may not allow accurate approximation of the original distances. HLM is proposed to leverage advantages of both schools. Under HLM, the clusters are mapped to separate Euclidean spaces, motivated by the belief that the local geometry of data is much simpler than the global. To account for complexity at the global level, discretization is then utilized.

HLM forms a smooth distribution by placing a smooth parametric distribution at each cluster centroid. This idea is inspired by the mixture model estimation for a distribution in the Euclidean space. The well-known classification expectation maximization (CEM) algorithm for estimating a mixture model iterates essentially a clustering step and a component-wise parameter fitting step. The EM algorithm can be viewed as performing soft clustering instead, while in the kernel density, every data point is treated as a cluster centroid. The technical difficulty for a general space is the lack of a parametric distribution. The basic assumption of HLM is that each cluster can be mapped to a distance-preserving Euclidean space and the mapped points follow a Gaussian distribution. It is theoretically difficult to validate this assumption for some well defined general spaces. In practice, we may not even have a mathematical specification of the space, so the assumption of HLM can only be taken as a computational mechanism. As a result, the mixture model constructed by HLM may not induce a proper probability measure on the original space. On the other hand, HLM is in line with the common practice in machine learning to analyze points mapped to a Euclidean space.

Consider a distance preserving mapping $\mathbb{M}$: $\Omega \to \mathcal{R}^d$, where $\mathcal{R}^d$ is the d-dimensional Euclidean space. Denote the distance on Ω by $D(\cdot, \cdot)$. Suppose Z is a random variable on Ω. Let a realization of Z be z.

For $\eta \in \Omega$, suppose $\mathbb{M}(\eta) = \mu$. We assume the mapped variable $X = \mathbb{M}(Z) \in \mathcal{R}^d$ follows a normal distribution $\mathcal{N}(\mu, \Sigma)$: μ is the mean and $\Sigma = \sigma^2 I$ is the covariance matrix, where I is the identity matrix.

Denote a Gamma distribution by $(\gamma : b, s)$, where b is the scale parameter and s is the shape parameter. Let $\Gamma(\cdot)$ be the Gamma function. The probability density function (pdf) of $(\gamma : b, s)$ is

$$f(u) = \frac{(\frac{u}{b})^{s-1} e^{-u/b}}{b\Gamma(s)}, \quad u \geq 0.$$

It is known that the squared Euclidean distance between X and the mean μ, $\|X - \mu\|^2$, follows a Gamma distribution $(\gamma : \frac{d}{2}, 2\sigma^2)$. The key idea of HLM is to estimate the parameters of the Gaussian distribution by estimating those of the Gamma distribution using only the squared distance $\|X-\mu\|^2$. Because the mapping $\mathbb{M}$ preserves the distance, $D^2(Z, \eta) = \|\mathbb{M}(Z) - \mathbb{M}(\eta)\|^2 = \|X - \mu\|^2$. Hence, the estimation can be based on distances in the original space, bypassing mapping ("hypothetical" mapping). Moreover, by estimating the Gamma distribution, the dimension d is determined. With b and s estimated, $d = 2s$ and $\sigma^2 = b/2$.

The pdf of X, $\varphi(x) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} e^{-\frac{\|x-\mu\|^2}{2\sigma^2}}$, can be recast with the Gamma distribution parameters b and s and the distance in the original space. Note that $x = \mathbb{M}(z)$.

$$\tilde{\varphi}(z) := \varphi(x) = \left(\frac{1}{\sqrt{\pi b}}\right)^{2s} e^{-\frac{D^2(z,\eta)}{b}}. \tag{1}$$

We call $\tilde{\varphi}(z)$ the *pseudo-density* function on Ω. It is a density for the mapping of Ω on $\mathcal{R}^d$, but is expressed in terms of the point $z \in \Omega$.

In the above discussion, a single Gaussian distribution is used to model the mapped data. We now impose a clustering structure on Ω. Consider a set of data points $\mathcal{G} = \{z_1, z_2, ..., z_N\}$, $z_i \in \Omega$. Suppose $\mathcal{G}$ is partitioned into M clusters $\mathcal{G}_j$ with corresponding centroid η_j, $j = 1, ..., M$. By HLM, cluster $\mathcal{G}_j$ and its centroid η_j can be mapped to $\mathcal{R}^d$ preserving the pairwise distance. Denote the mapping for $\mathcal{G}_j$ by $\mathbb{M}_j$. The mapping is performed separately for each cluster, but the mapped spaces have a common dimension d.

Under HLM, it is assumed that the mapped points of $\mathcal{G}_j$ can be modeled by a multivariate Gaussian distribution $\mathcal{N}(\mathbb{M}_j(\eta_j), \sigma_j^2 I)$ with mean equal to the mapped centroid $\mathbb{M}_j(\eta_j)$ and a spherical covariance. Then for $z_i \in \mathcal{G}_j$, $\|\mathbb{M}_j(z_i) - \mathbb{M}_j(\eta_j)\|^2$ are samples from Gamma distribution $(\gamma : s = \frac{d}{2}, b_j = 2\sigma_j^2)$. The pseudo-density of the jth cluster can be expressed by Equation (1). The pseudo-density on the overall Ω is then given by a mixture model, denoted by $\phi(z)$. We will from now on use cluster and component exchangeably since every mixture component is estimated using the data in one cluster. Let the prior probabilities for the components be ω_j, $j = 1, ..., M$, which are estimated by the empirical frequencies of the clusters. Then

$$\phi(z) = \sum_{j=1}^{M} \omega_j \left(\frac{1}{\sqrt{\pi b_j}}\right)^{2s} e^{-\frac{D^2(z,\eta_j)}{b_j}}. \tag{2}$$

Next, we describe how to estimate the Gamma distribution parameters s and b_j, $j = 1, ..., M$. Denote the set of indices for data points in $\mathcal{G}_j$ by $\mathcal{I}_j$, $j = 1, ..., M$. That is, if $i \in \mathcal{I}_j$, then $z_i \in \mathcal{G}_j$. Denote the

cardinality of $\mathcal{I}_j$ by $|\mathcal{I}_j|$. Obviously, $N = \sum_{j=1}^{M} |\mathcal{I}_j|$. We estimate the component prior ω_j by $|\mathcal{I}_j|/N$, $j = 1, ..., M$. If data point $z_i \in \mathcal{G}_j$, define $u_i = D^2(z_i, \eta_j)$, the squared distance to the corresponding centroid. Let the mean of the squared distances u_i, $i \in \mathcal{I}_j$, be $\bar{u}_j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} u_i$, and the overall mean $\bar{u} = \frac{1}{N} \sum_{i=1}^{N} u_i$.

Let $\psi(\cdot)$ be the di-gamma function [20]: for $s > 0$, $\psi(s) = \dfrac{d \log \Gamma(s)}{ds}$. It is shown that the maximum likelihood estimation for s and b_j, $j = 1, ..., M$, obeys the following equations, easily solvable numerically [4]:

$$\begin{cases} \log \hat{s} - \psi(\hat{s}) = \log \left[\prod_{j=1}^{M} \bar{u}_j^{|\mathcal{I}_j|/N} / (\prod_{i=1}^{N} u_i)^{1/N} \right] \\ \hat{b}_j = \bar{u}_j / \hat{s}, \quad j = 1, 2, ..., M \, . \end{cases} \tag{3}$$

It is assumed above that $u_i > 0$ for every i, which normally holds if the cluster centroid is different from any data point. However, in pairwise distance-based clustering, the centroid is often itself a data point. We thus exclude all the centroids from the data set once they have been determined. Moreover, in practice, we may obtain singleton cluster (with only one data point) due to limited data or outliers. To avoid zero distances, we remove all the singleton clusters from estimation. In addition, to increase the robustness of parameter estimation against small clusters, we shrink $\hat{b}_j$ toward a common value. We modify $\hat{b}_j = \bar{u}_j / \hat{s}$ slightly to $\hat{b}_j = \lambda \dfrac{\bar{u}_j}{\hat{s}} + (1 - \lambda) \dfrac{\bar{u}}{\hat{s}}$, where λ is a shrinkage factor. We set $\lambda = \frac{|\mathcal{I}_j|}{|\mathcal{I}_j| + 1}$, which imposes stronger shrinkage on smaller clusters. When $|\mathcal{I}_j| \to \infty$, the amount of shrinkage diminishes to zero. Finally, because the dimension of space $d = 2s$ should be an integer, we adjust the estimation $\hat{s}$ above to $s^* = \lfloor 2\hat{s} + 0.5 \rfloor / 2$.

In a more general scenario, the data points are assigned with weights w_i, $i = 1, ..., N$. Without loss of generality, assume normalized weights $\sum_{i=1}^{N} w_i = 1$. Define $\breve{u}_j = \dfrac{\sum_{i \in \mathcal{I}_j} w_i u_i}{\sum_{i \in \mathcal{I}_j} w_i}$ as the weighted mean of u_i's in cluster $\mathcal{G}_j$, $j = 1, ..., M$. We prove in Appendix A that the weighted maximum likelihood estimation for s and b_j, $j = 1, ..., M$, is solved by

$$\begin{cases} \log \hat{s} - \psi(\hat{s}) = \log \left[\prod_{j=1}^{M} \breve{u}_j^{\sum_{i \in \mathcal{I}_j} w_i} / \prod_{i=1}^{N} u_i^{w_i} \right] \\ \hat{b}_j = \breve{u}_j / \hat{s}, \quad j = 1, 2, ..., M \, . \end{cases} \tag{4}$$

Equations (3) and (4) are equivalent when $w_i = 1/N$ for all i.

2.2 Clustering Methods

Before HLM is applied to estimate mixture models, we first need to cluster data in each class and solve the cluster centroids. Distance-based clustering is a rich research topic with many algorithms developed. In our work, the following three methods are used, which we briefly explain.

Agglomerative Clustering (aka, linkage clustering) starts with every data point being treated as an individual cluster and merges recursively a pair of clusters with the minimum distance [21]. Different schemes are used to define the distance between a newly formed cluster and an existing one based on the latter's distances to the two clusters just merged. Suppose there are N data points. Then the computational

complexity is in the order $O(N^2)$, prohibitive for scaling up with large data sets.

Generalized k-means minimizes the total within cluster distance, in the same spirit as k-means, and is also referred to as k-medoids [22]. It can be regarded as a heuristic method for the well-known p-median problem in operation research [23]. As with k-means, the algorithm iterates the steps of cluster assignment and centroid update. The difference is that the cluster centroid is not the arithmetic mean of the data in a cluster because such a mean is not defined for general data. Instead, the data point with minimum total distance to all the other data points in the cluster is taken as the centroid. Suppose there are M clusters with roughly equal sizes. The computational complexity is in the order of $O(N \cdot M + N^2/M)$.

Vertex Substitution Heuristic (VSH) was first proposed by Teitz and Bart [24] as another heuristic approach to p-median problem. Similar to the generalized k-means, it first randomly selects several data points as the initial set of centroids and assigns the remaining data points to their closest centroids. Then in one sweep, every non-centroid data point is checked as a candidate for exchanging with a centroid. The exchange resulting in the largest reduction in the total within cluster distance is selected. This process is repeated until no further reduction can be found. The worst case complexity of VSH is $O(N^2)$. In practice, VSH is often fast and achieves more stable performance than generalized k-means [24].

3 Distance-based Classification via HLM

In this section, we present two approaches to classification based on HLM. The first approach, called *Mixture HLM* (M-HLM), performs clustering before estimating a mixture model via HLM. Here clustering plays the roles of a smoothing mechanism to suppress outliers and a data reduction mechanism to save computation. Experiments show that the smoothing effect of clustering can overshoot. Taking all the original data as centroids often outperforms a much reduced set of cluster centroids, albeit at the cost of more computation during testing. Our second approach, called *Kernel HLM* (K-HLM), yields a kernel version of the mixture model based on HLM without clustering. This approach is particularly appealing when the number of data points in a class is small, in which case clustering will likely cause the loss of valuable information.

3.1 Mixture HLM

Suppose there are K classes on Ω. M-HLM generates the classifier by the following steps.

1. Perform clustering on the data in each class $k = 1, ..., K$, by a distance-based method and identify the cluster centroids. Compute the distance between every data point and its corresponding centroid.

2. Introduce the following notations.

   - M_k, $k = 1, ..., K$: the number of clusters (aka, components) in each class.
   - $\bar{M} = \sum_{k=1}^{K} M_k$: the total number of clusters for all the classes.
   - $\mathcal{F}_k$: the set of indices for components in class k. $\mathcal{F}_k = \left\{ \breve{M}_k + 1, \breve{M}_k + 2, ..., \breve{M}_k + M_k \right\}$, where $\breve{M}_k = \sum_{k'=1}^{k-1} M_{k'}$, for $k > 1$, and $\breve{M}_1 = 0$.

- η_j, $j = 1, ..., \bar{M}$: the centroid of each cluster.
- ω_j, $j = 1, ..., \bar{M}$: the component prior probability with respect to its corresponding class.
- π_k, $k = 1, ..., K$: the prior probability of each class.

Estimate the component prior ω_j, $j = 1, ..., \bar{M}$, by the empirical frequency of the component with respect to its class. Also, estimate the class prior π_k, $k = 1, ..., K$, by their empirical frequencies.

3. Form a mixture model $\mathcal{M}_k$ for each class k. The pseudo-density for $Z \in \Omega$ under $\mathcal{M}_k$ is

$$\phi(z|\mathcal{M}_k) = \sum_{j \in \mathcal{F}_k} \omega_j \left(\frac{1}{\sqrt{\pi b_j}} \right)^{2s} e^{-\frac{D^2(z,\eta_j)}{b_j}}, \tag{5}$$

where b_j is the scale parameter for component j and s is the common shape parameter shared by all the components in all the classes. These parameters are estimated by Equation (3). It is in fact straightforward to let the shape parameter s vary with the class, that is, to have s_k. In the case of a common s, distances between data points and centroids in all the classes are used by Equation (3) to solve s, while for separate s_k's, only distances within the particular class k are used to estimate s_k.

4. To classify a test data point $z \in \Omega$, compute the posterior probability of z belonging to each class k:

$$p_k(z) = \frac{\pi_k \phi(z|\mathcal{M}_k)}{\sum_{l=1}^{K} \pi_l \phi(z|\mathcal{M}_l)}, \quad k = 1, 2, ..., K \ . \tag{6}$$

The class of z is then set to $\text{argmax}_{1 \le k \le K} \, p_k(z)$.

To determine the number of components M_k for every class k, we use cross validation. In our experiments, we set an equal number of components for each class for simplicity.

3.2 Kernel HLM

In this approach, a mixture model is formed by HLM without performing clustering, but instead in the manner of a kernel density estimate. Specifically, each training data point is treated as a cluster centroid in its class. Let the training data be $\{z_1, z_2, ..., z_N\}$. The number of data points in class k is n_k, $\sum_{k=1}^{K} n_k = N$. Without loss of generality, we assume the indices of the data in class k are $\{z_{\breve{n}_k+1}, ..., z_{\breve{n}_k+n_k}\}$, where $\breve{n}_k = \sum_{k'=1}^{k-1} n_{k'}$, for $k > 1$, and $\breve{n}_1 = 0$. We form a nonparametric model for class k:

$$\phi(z|\mathcal{M}_k) = \sum_{i=\breve{n}_k+1}^{\breve{n}_k+n_k} \frac{1}{n_k} \left(\frac{1}{\sqrt{\pi b}} \right)^{2s} e^{-\frac{D^2(z,z_i)}{b}},$$

where the scale parameter b and the shape parameter s of the Gamma distribution are obtained by the following data-driven method. For each data point z_i, we find its nearest neighbor in the same class, say $z_{i'}$. We then set $u_i = D^2(z_i, z_{i'})$, and model u_i, $i = 1, ..., N$, by the Gamma distribution $(\gamma : b, s)$.

Let $\bar{u} = \frac{1}{N}\sum_{i=1}^{N} u_i$, $\tilde{u} = (\prod_{i=1}^{N} u_i)^{1/N}$. The maximum likelihood estimator for s is solved by $\log \hat{s} - \psi(\hat{s}) = \log(\bar{u}/\tilde{u})$, where $\psi(\cdot)$ is the di-gamma function; and that for b is $\hat{b} = \bar{u}/\hat{s}$. In practice, we find this approach to choose kernel parameters effective. After the kernel pseudo-density for each class is obtained, a test sample is classified by maximizing the posterior probability in Equation (6).

Kernel HLM is similar to k-NN in the sense that both require distances to all the training data to classify a test point. In Kernel HLM, every training data point contributes smoothly to the decision function, while in k-NN, only the closest k neighbors matter.

4 Incremental Learning via HLM

Data streams are ordered sequences of data records that often arrive in batches at a high speed, or in bursts. They are common in the digital world, for instance, network traffic, financial transactions, web search, and social media feeds. It is desirable to train statistical models on the fly—to exploit available data immediately and to adapt efficiently with newly arrived data. Another scenario that motivates learning models incrementally is when the entire data cannot be loaded into memory at once. In this section, we will show that our proposed mixture models based on HLM can be estimated conveniently in an incremental learning setup. Two incremental learning schemes are proposed. The first is more efficient in computation, while the second attempts to achieve better clustering by combining old and new data.

4.1 Scheme I

The main idea of Scheme I is to cluster each batch of data separately and update the mixture model by adding new components. Let us follow the notation in Section 2.1 and Section 3. Recall that $\mathcal{I}_j$ is the set of indices for data in the jth cluster and u_i, $i = 1, ..., N$, is the squared distance between the ith data point and its corresponding centroid. Given any clustering result, to estimate the parameters in the mixture model, an inspection of Equation (3) reveals that we only need the following information about every cluster: the cluster size $|\mathcal{I}_j|$, the arithmetic mean of the u_i's with $i \in \mathcal{I}_j$, denoted by $\bar{u}_j$, and the geometric mean of all the u_i's, $\tilde{u} = (\prod_{i=1}^{N} u_i)^{1/N}$. Therefore, in incremental learning, data up to the current batch can be discarded after those statistics are stored. Only $\tilde{u}$ needs to be updated when new data arrive, while $|\mathcal{I}_j|$ and $\bar{u}_j$ for any existing jth cluster can simply be stored. With every new batch of data, more $|\mathcal{I}_j|$ and $\bar{u}_j$ will be computed and stored for the newly created clusters. Suppose the number of data that have arrived so far is N_1 and the number of data in a new batch is N_2. Let $N = N_1 + N_2$. Let $\tilde{u}_1 = (\prod_{i=1}^{N_1} u_i)^{1/N_1}$, which has been stored. When a new batch arrives, the geometric mean of all the squared distances can be updated by $\tilde{u} = \tilde{u}_1^{N_1/N}(\prod_{i=N_1+1}^{N} u_i)^{1/N}$, which does not require the old u_i's, $i = 1, ..., N_1$. The new statistics along with those computed from previous data are then combined to solve the parameters using Equation (3).

4.2 Scheme II

Although Scheme I is highly efficient in computation, it is a concern that clustering each batch of data separately degrades the quality of the clusters in comparison with clustering all the data together. This issue becomes more severe when an individual batch is small. We therefore propose Scheme II which trades off performance of clustering and computational intensity by sampling from the past data. To exploit clustering result on the past data, we draw samples per cluster and assign them weights so that data in clusters of different sizes are equally important. Then, weighted clustering is performed on randomly sampled points from the past data and all the data in the new batch.

We randomly sample a small number of data points in each cluster already obtained and assign them relatively large weights (assuming the cluster size is large). Smaller weights are given to data in the new batch because they are not subject to sampling. Specifically, in the past data, for a cluster j, if the cluster size $|\mathcal{I}_j|$ is larger than a pre-determined threshold v, we randomly select v samples as well as the cluster centroid from that cluster. The cluster centroid is taken because it is a natural representation for the whole cluster. If $|\mathcal{I}_j| \leq v$, all the samples from that cluster will be selected. In practice, v can be a small number. Suppose the ith sampled point is from the jth cluster. Assign weight w_i by

$$w_i = \begin{cases} \sqrt{\frac{|\mathcal{I}_j|+1}{v+1}} & \text{if } |\mathcal{I}_j| > v \\ 1 & \text{if } |\mathcal{I}_j| \leq v \ . \end{cases}$$

Larger weights are put on individual points from down sampled clusters with $|\mathcal{I}_j| > v$. On the other hand, the weight is dampened from a linear proportion in order to lower the influence of "old" data and to reduce the effect of decreased dispersion in sampled data. For data in the new batch, their weights are set to 1.

A weighted clustering algorithm is then applied to the sampled data and the data from the new batch. The three distance-based clustering algorithms introduced in Section 2.2 can be applied, taking a weighted distance matrix as input. Let the distance matrix $\mathbf{D} = (d_{i,i'})$ be the original symmetric matrix of distances between any pair of data points i and i', and $\mathbf{H}$ be the diagonal matrix with weights on the diagonal. The weighted distance matrix is defined by $\mathbf{R} = \mathbf{HD}$, no longer symmetric in general. We use the VSH clustering algorithm supplied with a weighted distance matrix [24]. Because the old cluster centroids have been assigned relatively large weights, they are likely to become new centroids again. In practice, we gradually increase the total number of clusters when new data arrive to boost the chance of new data becoming cluster centroids.

After the cluster centroids are obtained, we re-assign to their nearest centroids data points that are neither in the new batch nor among the sampled data. Thus all the data that have arrived so far are partitioned into the new clusters, although many did not participate in generating the centroids. The new clustering result is then used to estimate the parameters by Equation (3). This process is repeated at the arrival of every new batch. A possible way to save computation is to estimate the parameters using only the weighted samples and the new batch based on Equation (4).

5 Experiments

In this section, we compare our classification methods via HLM with several other distance-based classification methods on twelve data sets of wide variety. We first describe the experimental setup and data sets, and then present the classification results, computational time, and the performance of the incremental learning methods.

5.1 Methods and Setup for Comparison

The methods we have examined are described briefly below.

1. SVM: The standard SVM is applied in three ways. In the first version (SVM-similarities as kernel), the $N \times N$ symmetric distance matrix is modified into a kernel (positive semidefinite) by spectrum clip, which clips all the negative eigenvalues to zero [12]. In the other two versions (SVM-similarities as features), a feature vector is formed for each data point by its similarities to all the training data; then SVM with linear or Gaussian radial basis function (RBF) kernel is applied to those feature vectors.

2. Potential-SVM: This is a more generalized SVM applicable to any given $N \times N$ distance matrix which may not be square or positive definite [10].

3. SVM-KNN: A standard SVM is applied to classification using a kernel modified from the pairwise distance matrix for a test sample and its k nearest neighbors.

4. Similarity Discriminant Analysis (SDA): A group of generative modeling approaches to classification based on similarities (or distances) are considered, including the basic SDA, local SDA, and mixture SDA. Since local SDA consistently performs well in practice and is comparable to the other two [12], we select local SDA as the representative of this particular set of methods for comparison. Local SDA is reduced to k-NN if there is not enough data to fit distributions over the distances [12].

5. k-NN: Choose the class with the highest occurrences among a test sample's k nearest neighbors.

6. Weighted k-NN: The weight design can have two different aims: affinity (assign larger weights to the data points that are closer to the test sample) and diversity (assign smaller weights to highly similar data points). Specifically, three approaches of weight assignment are tested: affinity weights, kernel ridge interpolation weights (KRI), and kernel ridge regression weights (KRR) [12]. KRI-KNN and KRR-KNN consider both affinity and diversity.

7. Mixture HLM (vsh, gknn, agg) and Kernel HLM: Three clustering methods, VSH, generalized k-means, and agglomerative clustering, are used with Mixture HLM. The three algorithms are denoted in short by HLM (vsh), HLM (gknn), HLM (agg). Kernel HLM does not require clustering.

In Mixture HLM, we exclude clusters containing fewer than three data points from parameter estimation. The Ward's method [25] is used in agglomerative clustering. Generalized k-means algorithm is initialized

by the k-center algorithm which is also based on pairwise distances. For VSH, we select the clustering which yields the minimum total within cluster distance based on 20 random initializations.

For each data set, we randomly select 20% as test data and the remaining 80% as training. Parameters that need to be pre-chosen for the classifiers, such as the penalty parameter C and the RBF parameter γ for SVM, weight λ for KRI k-NN and KRR k-NN, the neighborhood size k for k-NN and local SDA, and the number of components in HLM, are all selected by 10-fold cross validation on the training set. The trained model is then applied to classify the held out test data. We repeat this process for 20 random partitions into training and test data. The"one-versus-one" scheme [26] is used in the multi-classification of SVM. We assume an equal number of components in each class for HLM; and the number of components is selected from $\{1, 2, 3, .., 10, 12, 14, 16\}$ by cross validation. For the other classifiers, the candidate values of parameters subject to cross validation selection are the same as those in [12] [1]. The only tuning parameter which users select for HLM is the number of components in each class, relatively simple compared with the other classifiers. For Kernel HLM, no tuning parameter is required.

5.2 Data Sets

Table 1: Summary of Data Sets

Name	# data	# classes	Symmetric	Vector	Distance Type
Amazon-47	204	47	No	No	Percentage
Aural Sonar	100	2	Yes	No	Human judgment
Caltech-101	8677	101	Yes	Yes	Kernel
Face Rec	945	139	Yes	No	Cosine similarity
Mirex07	3090	10	Yes	No	Human judgment
Patrol	241	8	No	No	Frequency
Voting	435	2	Yes	Yes	Value difference metric
Protein	213	4	Yes	No	Sequence-alignment similarity
Color Signature	600	6	Yes	No	Wasserstein distance
Photo Composition	150	3	Yes	No	IRM distance
Imagery	1400	5	Yes	Yes	Euclidean distance
Sonar	208	2	Yes	Yes	Euclidean distance

Table 1 presents the basic information about the data sets, e.g., data size, the number of classes. The column entitled "Symmetric" shows whether the distance is symmetric, and the column entitled "Vector" shows whether the data object is a vector. For details on the first eight data sets, we refer to [12]. We add four more new data sets described below.

Imagery data contains 1400 images from 5 semantic classes: mountain scenery (300), women (300), flower (300), city scene (300), and beach scene (200). We indicate the class size in the parenthesis. Each image is represented by a 64 dimensional feature vector and the Euclidean distance is used.

Color signature contains 600 images each represented by its color signature, which is a set of weighted

[1] See Table 2 in [12] for details on the ranges of parameters.

vectors (or discrete distribution). There are 6 classes in total, 5 of which are the classes in Imagery data and the last one is a new semantic class named "man-made items". Every class includes 100 images. The color signature of an image is formed based on its segmentation. The average color vector of the pixels in each segment and the percentage of pixels in the segment with respect to the whole image are recorded in the color signature. The number of segments is dynamically determined, the average for this data set being 10. The distance between color signatures is the Kantorovich-Wasserstein distance [27], better known as the Mallows distance in statistics [28].

Photography composition is used in [29] as a benchmark data set for composition classification. There are 150 photos equally divided into three classes: horizontal, vertical, and centered. The spatial layout of each photo is represented by a set of weighted spatial relational vectors (SRV). We refer to [29] for details on SRV. The distance used is the integrated region matching (IRM) distance [30], a greedy variant of the Wasserstein distance.

Sonar data is from the UCI machine learning repository, with 208 samples from two classes (111, 97). The data are 60-dimensional vectors under the Euclidean distance.

Because HLM takes distances as input, for some data sets given with similarities, we need to convert similarities into dissimilarities, aka, distances. If the similarity S has an obvious upper bound $\bar{S}$, the corresponding distance is defined as $\bar{S} - S$, otherwise, $1/S$. On the other hand, for the algorithms taking similarities as input, we also need to convert distances into similarities. Since distances have no upper bound, the corresponding similarities are defined as their reciprocals. We use an appropriate upper bound of all the similarities as the self-similarity of a data point. In addition, if the distance between two data points, $D(z, z')$, is not symmetric, for example, those for the Amazon-47 and Patrol data, we use the symmetrized distance $[D(z, z') + D(z', z)]/2$.

The Amazon-47, Face Rec, and Patrol data sets are so small that it is not meaningful to perform clustering in each class. Among the HLM-based algorithms, we thus only apply Kernel HLM to these data sets. For the data set Patrol, the distances between data points only take three values: 0.0, 0.5, and 1.0. The distance between a data point and its nearest neighbor is often 0.0, causing numerical issues for parameter estimation in Kernel HLM. Therefore, for Patrol, we manually select the shape and scale parameters from the range of estimated values based upon HLM (vsh).

5.3 Classification Results

We have tested fifteen classification algorithms. These algorithms fall into several categories: k-NN and its variations, SVM-based, local SDA, and HLM-based. For clarity, we present results for six representative algorithms in Table 2. For completeness, we report the results of the other nine algorithms in Table 3. We choose one algorithm from each of the three categories in existing literature, namely, k-NN, local SDA, and SVM-similarities as features (rbf). These algorithms are well-known and basic among their respective categories. In addition, the more complicated versions are not evidently stronger. For the newly developed Mixture HLM algorithms using clustering, we found that HLM (vsh) consistently works well while HLM

(gknn) exhibits more variation in performance. HLM (agg) is also relatively stable in performance but is more expensive in computation. We thus show results of HLM (vsh) and Kernel HLM in Table 2.

Because the results of HLM (vsh) and Kernel HLM differ considerably on several data sets with either outperforming the other in some cases, one may naturally raise the question which algorithm should be used in practice. Can we reliably select the better of the two algorithms during training? We propose a solution by cross validation. As described in Section 5.1, cross validation is used to select the number of components per class for HLM (vsh). We simply augment the candidate pool for the number of components by n_k, the total data size in the kth class. The case of n_k corresponds to Kernel HLM. We denote this meta-algorithm by HLM (kernel-vsh) and also report its classification results in Table 2.

Table 2: Classification error rates of distance-based classifiers (I)

Classifier	Amazon-47	Aural Sonar	Caltech-101	Color-signature
k-NN	16.95 (4.85)	17.25 (6.80)	41.55 (0.95)	33.25 (3.65)
Local SDA	16.83 (5.11)	17.75 (7.66)	41.99 (0.52)	35.71 (2.67)
SVM-similarities as features (rbf)	82.68 (5.92)	**14.00 (7.18)**	38.16 (0.75)	36.58 (3.54)
Kernel HLM	**15.61 (5.37)**	**13.75 (5.67)**	40.18 (1.00)	**31.54 (3.96)**
HLM (vsh)	NA	16.00 (5.39)	48.52 (1.08)	36.42 (3.42)
HLM (kernel-vsh)	NA	**15.25 (4.60)**	40.18 (1.00)	32.92 (5.06)
Classifier	Face Rec	Imagery	Mirex	Patrol
k-NN	4.23 (1.43)	36.64 (3.04)	61.21 (1.97)	**11.88 (4.42)**
Local SDA	4.55 (1.67)	42.87 (2.52)	60.94 (1.94)	**11.77 (4.62)**
SVM-similarities as features (rbf)	13.97 (2.08)	47.16 (2.38)	**55.72 (2.06)**	21.98 (4.86)
Kernel HLM	**3.81 (1.36)**	36.64 (3.13)	69.42 (2.33)	**11.46 (4.09)**
HLM (vsh)	NA	44.73 (2.71)	61.62 (1.92)	NA
HLM (kernel-vsh)	NA	36.64 (3.13)	61.62 (1.92)	NA
Classifier	Protein	Photo Composition	Sonar	Voting
k-NN	29.88 (9.82)	24.83 (6.79)	**20.71 (6.48)**	5.52 (1.61)
Local SDA	17.44 (6.52)	22.67 (8.92)	**20.00 (6.79)**	6.38 (2.07)
SVM-similarities as features (rbf)	**1.05 (1.37)**	**19.67 (7.14)**	**21.31 (5.71)**	6.03 (1.85)
Kernel HLM	29.07 (7.52)	23.17 (8.66)	23.81 (5.05)	6.09 (1.86)
HLM (vsh)	23.49 (10.55)	23.50 (7.11)	24.40 (4.69)	**4.89 (2.12)**
HLM (kernel-vsh)	24.42 (9.91)	23.50 (7.11)	23.57 (5.42)	**4.89 (2.12)**

In Table 2 and 3, for each method, we show its mean error rate in percent and standard deviation (values in parenthesis) across 20 random partitions of training and test data. The one-sided Wilcoxon signed-rank test at the significance level 0.05 is used to test whether the result of one method, in terms of the error rates obtained from the 20 random partitions, differs significantly from another, as is done in [12]. The lowest classification error rate in each column of Table 2 is underscored. For each data set, we show in bold font the lowest error rate among all the methods (that is, the best across results in both Table 2 and Table 3). In order to mark the methods that yield error rates not significantly worse than the best result according to the Wilcoxon test, we also show in bold font the error rates of these methods.

According to Table 2, Kernel HLM ranks on the top most frequently. Among the algorithms reported

Table 3: Classification error rates of distance-based classifiers (II)

Classifier	Amazon-47	Aural Sonar	Caltech-101	Color-signature
affinity k-NN	**15.24 (5.05)**	15.50 (5.89)	39.20 (0.86)	32.21 (3.12)
KRI k-NN (clip)	17.44 (4.89)	**13.75 (6.87)**	30.13 (0.42)	**31.13 (3.01)**
KRR k-NN (pinv)	**16.10 (4.90)**	15.50 (6.69)	**29.90 (0.44)**	31.79 (3.36)
SVM-KNN (clip)	17.56 (4.60)	**13.75 (7.40)**	36.82 (0.60)	31.25 (3.45)
SVM-similarities as kernel (clip)	76.83 (7.85)	**13.00 (5.34)**	33.49 (0.78)	34.96 (3.91)
SVM-similarities as features (linear)	74.76 (9.10)	**14.25 (6.94)**	38.18 (0.78)	36.71 (3.59)
P-SVM	70.12 (8.82)	**14.25 (5.97)**	34.23 (0.95)	33.54 (4.06)
HLM (gknn)	NA	**14.50 (5.22)**	52.01 (1.13)	39.79 (4.82)
HLM (agg)	NA	**13.75 (6.30)**	48.03 (1.29)	38.42 (4.04)
Classifier	Face Rec	Imagery	Mirex	Patrol
affinity k-NN	4.21 (1.40)	35.98 (3.63)	61.15 (1.90)	**11.56 (4.03)**
KRI k-NN (clip)	4.18 (1.38)	**35.43 (3.32)**	61.20 (2.03)	**11.56 (4.54)**
KRR k-NN (pinv)	4.10 (1.40)	**35.91 (3.54)**	61.18 (1.96)	12.81 (4.62)
SVM-KNN (clip)	4.23 (1.25)	36.13 (3.60)	61.25 (1.95)	**11.98 (4.36)**
SVM-similarities as kernel (clip)	4.18 (1.25)	44.23 (2.87)	57.83 (2.05)	25.63 (6.01)
SVM-similarities as features (linear)	4.39 (1.31)	48.86 (2.59)	**55.54 (2.52)**	23.44 (5.78)
P-SVM	4.05 (1.44)	40.93 (2.33)	63.81 (2.70)	40.42 (5.94)
HLM (gknn)	NA	39.79 (4.82)	81.10 (1.66)	NA
HLM (agg)	NA	46.27 (2.98)	61.54 (1.74)	NA
Classifier	Protein	Photo Composition	Sonar	Voting
affinity k-NN	30.81 (6.61)	25.33 (7.56)	**20.36 (6.00)**	5.86 (1.78)
KRI k-NN (clip)	30.35 (9.71)	22.83 (8.18)	**20.00 (4.90)**	**5.29 (1.80)**
KRR k-NN (pinv)	9.65 (4.95)	22.67 (8.79)	**20.71 (5.22)**	**5.52 (1.69)**
SVM-KNN (clip)	11.86 (5.50)	21.50 (9.57)	**20.00 (5.29)**	**5.17 (2.20)**
SVM-similarities as kernel (clip)	5.35 (4.60)	**19.33 (7.64)**	**19.29 (6.25)**	**4.89 (2.05)**
SVM-similarities as features (linear)	3.02 (2.76)	**19.33 (7.42)**	**20.60 (5.60)**	**5.34 (1.86)**
P-SVM	1.86 (1.89)	NA	**19.29 (4.64)**	**5.34 (1.72)**
HLM (gknn)	14.53 (12.59)	23.67 (8.02)	31.43 (5.81)	6.95 (2.67)
HLM (agg)	15.00 (13.00)	24.00 (8.86)	24.88 (6.18)	**5.17 (2.37)**

in this table, Kernel HLM performs best on 6 data sets, while SVM (rbf) performs best on 4 data sets and k-NN on 1 data set (tied best with Kernel HLM on Imagery). If we restrict the comparison to the three generative modeling approaches, local SDA, HLM (vsh), and Kernel HLM, Kernel HLM performs best on 7 data sets, while local SDA performs best on 4 data sets and HLM (vsh) on 1 data set. Among all the classification results in both Table 2 and Table 3, Kernel HLM achieves the lowest error rate or an error rate not significantly worse than the lowest on 5 data sets, tied with SVM (rbf).

Which of the two, Kernel HLM or HLM (vsh), yields a higher accuracy on a particular data set can only be known after testing, while HLM (kernel-vsh) makes the choice beforehand in training. Among the 9 data sets to which HLM (kernel-vsh) is applied, HLM (kernel-vsh) chooses consistently across all the random partitions the stronger one out of Kernel HLM and HLM (vsh) for 4 data sets: Caltech-101, Imagery, Mirex, Voting. Consequently, HLM (kernel-vsh) yields the same result as either Kernel HLM or HLM (vsh), whichever is better, on these data sets. On Photo Composition data, HLM (kernel-vsh) consistently

chooses HLM (vsh) although HLM (vsh) performs slightly worse on average than Kernel HLM. However, on this data set, the difference in the results of HLM (vsh) and Kernel HLM is quite small, 23.50% versus 23.17%. On Sonar, HLM (kernel-vsh) performs slightly better than both Kernel HLM and HLM (vsh). On Aural Sonar, Color-signature, and Protein, the accuracy of HLM (kernel-vsh) is between Kernel HLM and HLM (vsh), and most frequently, closer to the better of the two. To summarize, the results show that cross validation is a reliable way to choose between Kernel HLM and HLM (vsh); and the meta-algorithm HLM (kernel-vsh) performs nearly as well as the better one of Kernel HLM and HLM (vsh) chosen in hindsight on any particular data set.

The performance of SVM (rbf) deviates remarkably from the other algorithms on several data sets. For Protein data, it yields a significantly lower error rate than the others, 1.05% versus other values ranging from 17.44% to 29.88%. On the other hand, it performs much worse than the others on Amazon-47 and Patrol. For Amazon-47, SVM (rbf) has an error rate of 82.68% while the others range from 15.61% to 16.95%; for Patrol, its error rate is 21.98% while the others are tightly around 11.50%. Similar performance on these three data sets, either very good or very poor, is also observed for SVM (clip), SVM (linear), and P-SVM, as shown in Table 3. In a sense, the SVM-based algorithms are more volatile.

When the input data are vectors, it is interesting to investigate the effect on classification by using the distance/similarity alone and ignoring the algebraic aspect of the data. For the vector-type Sonar and Imagery data, we apply the standard SVM with linear and Gaussian RBF kernels to the original vectors. The penalty parameter C and RBF kernel parameter γ are selected by cross validation using a smaller set of candidate values than those in [12] due to the excessive running time to tune the parameters. The experiments are conducted under the aforementioned experimental setup. For Sonar, the mean error rates (standard deviations) of SVM (standard linear) and SVM (standard rbf) are 16.90% (5.05%) and 20.00% (5.40%) respectively; for Imagery, the corresponding error rates are 37.88% (3.29%) and 32.13% (2.27%). Compared with the classification methods listed in Table 2 and Table 3, we see that SVM (standard linear) achieves the lowest error rate for Sonar while SVM (standard rbf) achieves the lowest error rate for Imagery. These two lowest error rates are also statistically significant among all the classification results. On the other hand, the worse result of the two methods on either data set is outperformed by a distance-based classification method. In addition, there is statistically significant difference between the classification results of SVM (standard linear) and SVM (standard rbf) on both data sets, while no such significance is observed between SVM (linear) and SVM (rbf) using similarities as features. This indicates that the particular geometry of the data in the vector space has played an important role. Even when SVM is applied in the original space, a specific kernel has to be used to leverage the geometry.

5.4 Computational Complexity and Running Time

Recall that the total number of data is $N = \sum_k n_k$, where n_k is the number of data in class k, and the total number of clusters for all the classes is $\bar{M} = \sum_k M_k$, where M_k is the number of cluster components in class k. The worst scenario complexity of the three distance-based clustering algorithms (vsh, gknn,

agg) is $\sum_k n_k^2$. After distances between all the data points and their corresponding centroids are obtained, quantities required to set up Equation (3) can be computed in O(N) time. We solve Equation (3) numerically by binary search over a fixed range, which is of complexity O($c\bar{M}$), where c is a constant time to solve a single equation. Therefore, provided with the clustering result, the complexity of estimating mixture models for all the classes is $O(N + c\bar{M})$. Since $N > \bar{M}$, the complexity of model estimation is linear in the total number of data N. The main computational cost is thus on clustering.

In practice, we find that VSH runs fast and returns good clustering results. Compared with local SDA and SVM-based classifiers, HLM (vsh) has significantly shorter running time, making it attractive for large scale computation. The two parameters of SVM (rbf), the penalty parameter C and the RBF parameter γ, are selected by grid search using cross validation, an extremely computationally intensive process. In contrast, HLM (vsh) only has one tuning parameter, the number of components in a class, to choose by cross validation. For SVM (rbf), we use the "one-versus-one" scheme for multi-classification. Crammer and Singer [31] proposed a structural multiclass SVM which reduces running time significantly by separating all K classes together. When similarities are treated as features, such structural SVM can be applied directly to cut running time. However, for other distance schemes in distance-based classification using SVM, for instance, in the case of modifying the pairwise distance matrix into a special kernel, how to apply structural multiclass SVM needs future investigation, which is beyond the scope of this paper.

Kernel HLM has achieved highly competitive classification performance on most data sets. It is similar to k-NN in that the computation during classification is mostly on computing the distances between a test point and every training point. During training, k-NN is not totally free in computation because cross validation is used to choose k, while Kernel HLM only estimates two parameters by an extremely fast algorithm after one round of 1-NN is performed. However, compared with HLM (vsh) exploiting a possibly much smaller set of cluster centroids than the original data, Kernel HLM is expensive during classification. For large data sets, a subset of data points may be sampled from each class and used as training data in Kernel HLM. This approach is interesting to investigate in the future.

Table 4: Running time of distance-based classifiers

Classifier	Amazon-47	Color-signature	Imagery	Mirex
k-NN	50 ms	50 ms	400 ms	950 ms
Local SDA	4.7 s	2 min 2.9 s	4 min 33.6 s	23 min 26.9 s
SVM-similarities as features (rbf)	10.5 s	2 min 21.3 s	20 min 4.8 s	220 min 16.8 s
HLM (vsh)	NA	40.2 s	4 min 47.8 s	11 min 1.3 s
Kernel HLM	30 ms	280 ms	1.7 s	4.4 s

In Table 4, comparison of the running time is provided for k-NN, local SDA, SVM (rbf), HLM (vsh), and Kernel HLM on several example data sets. The experiments run on a 2.66 GHz Intel CPU. HLM-based algorithms are implemented in C, local SDA is in Matlab and C/MEX [2], and the remaining algorithms are

[2]http://staff.washington.edu/lucagc/software.html. We use the Matlab Executable (MEX) files.

in C++ [3]. Table 4 reports the average running time to finish one round of training and testing, the average taking over 20 random partitions into training and test data. As Table 4 shows, HLM (vsh) and Kernel HLM are significantly faster than local SDA and SVM (rbf) on all the data sets except Imagery for which HLM (vsh) and local SDA are close in speed. For all the data sets, SVM (rbf) has the longest running time while k-NN has the shortest. On some data sets, the running time of SVM (rbf) is larger than that of Kernel HLM or k-NN by more than three orders of magnitude. Compared with k-NN, Kernel HLM is within one order of magnitude slower, caused mainly by the cost of estimating the scale and shape parameters. For local SDA, SVM (rbf), and HLM (vsh), the main computational cost is on training, while the classification of test data is fast. Both Kernel HLM and k-NN are computationally intensive for classifying test data if the training data set is large. For example, on Mirex data, the average testing time of Kernel HLM across 20 random partitions is 295 ms while HLM (vsh) takes only 29 ms.

5.5 Incremental Learning Results

Given a data set, we randomly select 20% of the data as the held-out test and the remaining 80% as training data in incremental learning. Among the training data, 20% are randomly selected as the initial training batch, and the rest are divided into eight batches of equal size (10% each). One round of incremental learning is performed on every newly arrived batch. We experiment with the two schemes of HLM-based incremental learning introduced in Section 4. As a baseline comparison, the performance of the trivial batch learning method, which retrains all the data that have arrived so far are also reported. VSH is used to perform clustering for all the methods. Specifically, in HLM incremental learning scheme (II), VSH performs the weighted clustering by taking weighted distance matrix defined in Section 4.2.

For both of the HLM incremental learning schemes, the number of components in each class in the initial training batch is set to 4. Because the size of every new batch is comparable with the initial data, in HLM incremental learning Scheme (I), we set the number of components in each class for every new batch to 4 as well. In HLM incremental learning Scheme (II), to prevent old cluster centroids from staying as the only centroids, we gradually increase the number of components in each class by 2 at every learning round. To establish a common ground for comparing this scheme and the baseline retraining scheme, we use the same number of components in each class at every learning round for the two approaches.

Table 5: Running time of HLM-based incremental learning methods in seconds

Classifier	Color signature	Imagery	Mirex	Voting
HLM-Incremental (I)	0.22	0.40	0.84	0.14
HLM-Incremental (II)	1.06	2.48	2.87	0.59
HLM-Retrain	1.72	12.54	22.89	2.04

When the available training data increase, the corresponding classification error rates achieved by the

[3] http://ee.washington.edu/research/guptalab/similaritylearning/simMLL-linux.tgz.

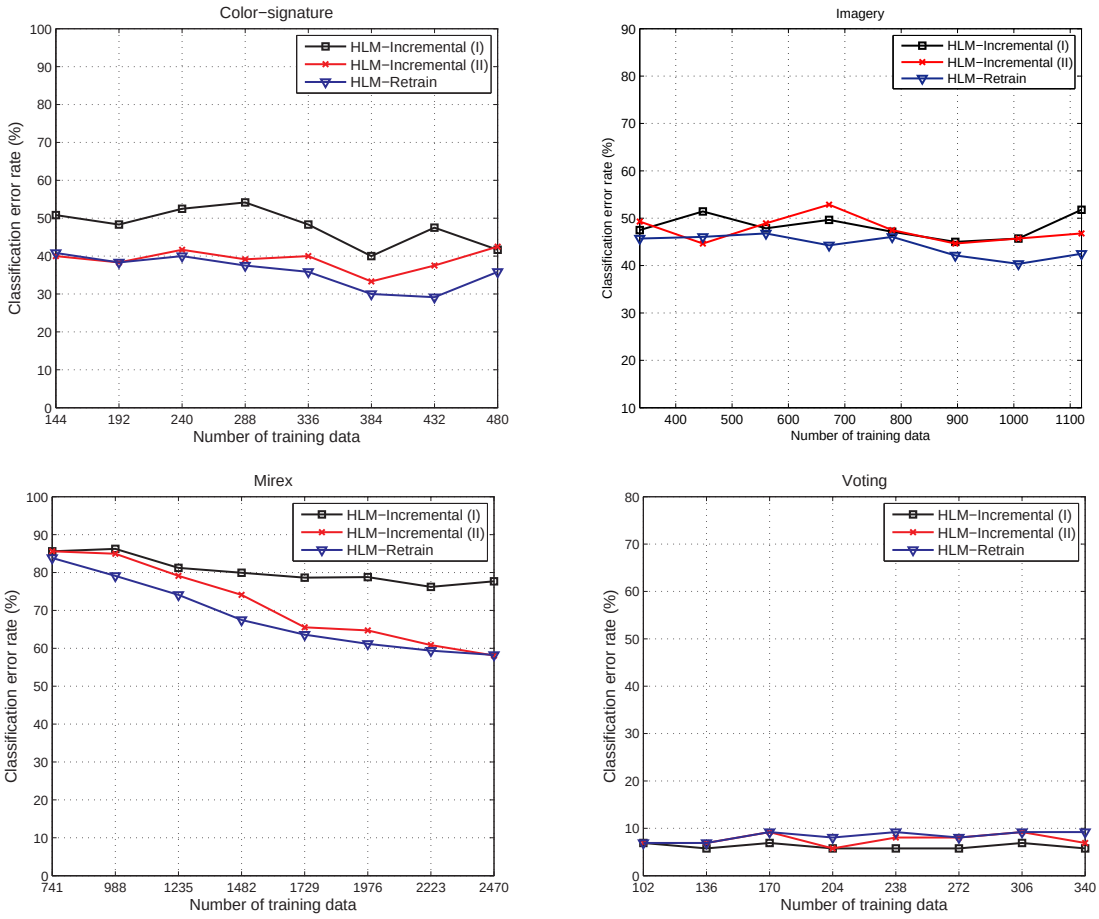

Figure 1: Compare the two HLM incremental learning methods versus the baseline of retraining on all the available data using four data sets. The classification error rates are shown at every learning round when the available training data increase.

methods are shown in Figure 1. We denote the two incremental learning methods by HLM-Incremental (I) and (II), and the baseline scheme by HLM-Retrain. In HLM-Incremental (II), the threshold v for down sampling in any cluster is 4. Again, clusters containing fewer than three data points are not used in the parameter estimation of HLM. All the experiments are conducted on a 2.66 GHz Intel CPU. Table 5 presents the total running time across all the learning rounds for the three methods. HLM-Incremental (I) has the shortest running time on all the data sets, while HLM-Incremental (II) is the second fastest, and HLM-Retrain is the slowest. The reduction in computation time achieved by HLM-Incremental (I)/(II) with respect to HLM-Retrain is proportionally most prominent on the largest data set, Mirex.

As shown in Figure 1, for Mirex data and Color signature data, except for the last two learning rounds, the classification error rates follow a clear decreasing trend at the increase of available training data. For these two data sets, HLM-Retrain stays as the winner for most of the learning rounds. However, it is interesting to note that for Voting data, the fastest scheme HLM-Incremental (I) is the winner among the three algorithms across all the rounds; and HLM-Incremental (II) outperforms HLM-Retrain at several rounds. For Mirex and Color signature, HLM-Incremental (II), the second fastest scheme, performs slightly worse

than HLM-Retrain at most of the learning rounds; and the fastest scheme HLM-Incremental (I) performs worst at almost all the learning rounds. This observation demonstrates a trade-off between computational intensity and performance. For all the data sets except Color signature, less than 5% difference is observed between the classification error rates obtained by HLM-Incremental (II) and HLM-Retrain at the last learning round. HLM-Incremental (II) even performs slightly better than HLM-Retrain on Mirex and Voting data at the last learning round. For Imagery and Voting data, the error rate curves for all the methods fluctuate, and at the last learning round, the error rates are only slightly better or even worse than where they start off after the first round. The fluctuations of the error rate curves are also observed at the last two learning rounds for Color signature. We believe that such counter intuitive outcomes have resulted from the inherent randomness of the data.

6 Conclusions

A distance-based mixture modeling approach based on hypothetical local mapping (HLM) is proposed. Because only pairwise distances are needed, HLM is particularly useful for data that cannot be easily described by a mathematical entity. The application of the proposed mixture model to classification is explored. We have compared this approach with several other state-of-the-art distance-based classification methods on various data sets. Experimental results show that HLM-based algorithms perform competitively at low computational cost during both training and testing. Because a mixture model is estimated for each class separately, scalability in the number of classes is achieved. HLM adapts readily to learning a classifier in an incremental manner. We have proposed two incremental learning schemes for HLM and found that at a much faster speed, they achieve similar classification accuracy as the baseline of retraining over all the available data.

Acknowledgement

The authors would like to thank the reviewers for their valuable comments and suggestions. This work is supported partly by the National Science Foundation under the grant CCF-0936948.

A Proof of Equation (4)

Recall N is the data size, M is the number of clusters, $\mathcal{I}_j$ is the data index set for the jth cluster. Suppose the squared distances between the data points and their centroids are u_i with weights w_i, $i = 1, ..., N$. We have $\sum_{i=1}^{N} w_i = 1$. Denote $\mathbf{u} = (u_1, u_2, ..., u_N)$ and $\mathbf{w} = (w_1, w_2, ..., w_N)$. The maximum likelihood

estimator maximizes the following weighted log likelihood:

$$\begin{aligned} L(\mathbf{u}|s, b_1, b_2, ..., b_M) &= \sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \log f(u_i) \\ &= \sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right] . \end{aligned} \tag{7}$$

With a fixed s, $L(\mathbf{u}|s, b_1, b_2, ..., b_M)$ can be maximized individually on every b_j:

$$\max_{b_1,...,b_M} L(\mathbf{u}|s, b_1, b_2, ..., b_M) = \sum_{j=1}^{M} \max_{b_j} \sum_{i \in \mathcal{I}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right] . \tag{8}$$

Because $\sum_{i \in \mathcal{I}_j} w_i \left[(s-1) \log u_i - s \log b_j - \frac{u_i}{b_j} - \log \Gamma(s) \right]$ is a concave function of b_j, we can obtain its maximum by setting the first derivative to zero: $\sum_{i \in \mathcal{I}_j} w_i (-\frac{s}{b_j} + \frac{u_i}{b_j^2}) = 0$. Let $\breve{u}_j = \frac{\sum_{i \in \mathcal{I}_j} w_i u_i}{\sum_{i \in \mathcal{I}_j} w_i}$ be the weighted average squared distance for centroid j. Then b_j is solved by $b_j = \frac{\breve{u}_j}{s}$. Now substitute b_j into (8):

$$\max_s L(\mathbf{u}|s) = \max_s \sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \left[s \log s + s \cdot (\log \frac{u_i}{\breve{u}_j} - \frac{u_i}{\breve{u}_j}) - \log \Gamma(s) - \log u_i \right] .$$

Since $\log \Gamma(s)$ is a convex function of s, it is easy to show that $L(\mathbf{u}|s)$ is a concave function of s. The maximum of $L(\mathbf{u}|s)$ is thus determined by setting its first derivative to zero:

$$\sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \log s + \sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \log \frac{u_i}{\breve{u}_j} - \sum_{j=1}^{M} \sum_{i \in \mathcal{I}_j} w_i \psi(s) = 0 ,$$

which is equivalent to:

$$\log \hat{s} - \psi(\hat{s}) = \log \left[\frac{\prod_{j=1}^{M} \breve{u}_j^{\sum_{i \in \mathcal{I}_j} w_i}}{\prod_{i=1}^{N} u_i^{w_i}} \right] . \tag{9}$$

References

[1] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. and Machine Intel.*, vol. 24(4), pp. 509-522, 2002.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215(3), pp. 403-410, 1990.

[3] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures," in *Proc. ACM MM*, 2006, pp. 911-920.

[4] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures," *IEEE Trans. Pattern Anal. and Machine Intel.*, vol. 30(6), pp. 985-1002, 2008.

[5] L. Cazzanti and M. R. Gupta, "Local similarity discriminant analysis," in *Proc. ICML*, 2007, pp. 137-144.

[6] L. Cazzanti, M. R. Gupta, and A. J. Koppal, "Generative models for similarity-based classification," *Pattern Recogn.*, vol. 41(7), pp. 2289-2297, 2008.

[7] L. Cazzanti, "Generative models for similarity-based classification," *Ph.D. Thesis*, Department of Electrical Engineering, University of Washington, 2007.

[8] L. Cazzanti, S. Feldman, M. R. Gupta, M. Gabbay, "Multi-task regularization of generative similarity models," in *Proc. SIMBAD*, 2011, pp. 90-103.

[9] S. Feldman, B. A. Frigyk, M. R. Gupta, L. Cazzanti, and P. Sadowski. (2012, Aug.). Multi-task output space regularization. [Online]. Available: *http://arxiv.org/abs/1107.4390v3*

[10] S. Hochreiter and K. Obermayer, "Support vector machines for dyadic data," *Neural Comput.*, vol. 18(6), pp. 1472-1510, 2006.

[11] H. Zhang, A. C. Berg , M. Maire, and J. Malik, "SVM-KNN: discriminative nearest neighbor classification for visual category recognition," in *Proc. IEEE CVPR*, 2006, pp. 2126-2136.

[12] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based classification: concepts and algorithms," *J. Mach. Learn. Res.*, vol. 10, pp. 747-776, 2009.

[13] E. Pekalska, P. Paclic, and R. P.W. Duin, "A generalized kernel approach to dissimilarity-based classification," *J. Mach. Learn. Res.*, vol. 2, pp. 175-211, 2001.

[14] P. Simard, Y. L. Cun, and J. Denker, "Efficient pattern recognition using a new transformation distance," *Adv. Neural Inform. Process. Syst.*, vol. 5, pp. 50-68, 1993.

[15] D. Weinshall, D.W. Jacobs, and Y. Gdalyahu, "Classification in non-metric spaces," *Adv. Neural Inform. Process. Syst.*, vol. 11, pp. 838-844, 1999.

[16] S. Cost and S. Salzberg, "A weighted nearest neighbor algorithm for learning with symbolic features," *Mach. Learn.*, vol. 10(1), pp. 57-78, 1993.

[17] M. R. Gupta, R. M. Gray, and R. A. Olshen, "Nonparametric supervised learning by linear interpolation with maximum entropy," *IEEE Trans. Pattern Anal. and Machine Intel.*, vol. 28(5), pp. 766-781, 2006.

[18] T. Graepel, R. Herbrich, and K. Obermayer, "Classification on pairwise proximity data," *Adv. Neural Inform. Process. Syst.*, vol. 11, pp. 438-444, 1999.

[19] R. P.W. Duin, E. Pekalska, and D. de Ridder, "Relational discriminant analysis," *Pattern Recogn. Lett.*, vol. 20, pp. 1175-1181, 1999.

[20] M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions, 3rd ed.*, New York: Wiley, 2000.

[21] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31(3), pp. 264-323, 1999.

[22] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. New York: Springer-Verlag, 2001.

[23] F. E. Maranzana, "On the location of supply points to minimize transport costs," *Oper. Res. Quart.*, vol. 15, pp. 261-270, 1964.

[24] M. B. Teitz and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," *Oper. Res.*, vol. 16, pp. 955-961, 1968.

[25] J. H. Ward, "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58(301), pp. 236-244, 1963.

[26] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13(2), pp. 415-425, 2002.

[27] S. T. Rachev, "The Monge-Kantorovich mass transference problem and its stochastic applications," *Theory Probab. Appl.*, vol. 29(4), pp. 647-676, 1985.

[28] C. L. Mallows, "A note on asymptotic joint normality," *Ann. Math. Stat.*, vol. 43(2), pp. 508-515, 1972.

[29] L. Yao, P. Suryanarayan, M. Qiao, J. Z. Wang and J. Li, "OSCAR: On-site composition and aesthetics feedback through exemplars for photographers," *Int. J. Comput. Vision* , vol. 96(3), pp. 353-383, 2012.

[30] J. Li, J. Z. Wang, and G. Wiederhold, "IRM: integrated region matching for image retrieval," in *Proc. ACM MM*, 2000, pp. 147-156.

[31] K. Crammer and Y. Singer, "On the Algorithmic Implementation of Multi-class SVMs," *J. Mach. Learn. Res.*, vol. 2, pp. 265-292, 2001.