Simplifying Mixture Models Through Function Approximation

Kai Zhang and James T. Kwok, *Senior Member, IEEE*

***Abstract*—The finite mixture model is widely used in various statistical learning problems. However, the model obtained may contain a large number of components, making it inefficient in practical applications. In this paper, we propose to simplify the mixture model by minimizing an upper bound of the approximation error between the original and the simplified model, under the use of the L_2 distance measure. This is achieved by first grouping similar components together and then performing local fitting through function approximation. The simplified model obtained can then be used as a replacement of the original model to speed up various algorithms involving mixture models during training (e.g., Bayesian filtering, belief propagation) and testing [e.g., kernel density estimation, support vector machine (SVM) testing]. Encouraging results are observed in the experiments on density estimation, clustering-based image segmentation, and simplification of SVM decision functions.**

***Index Terms*—Clustering, mixture models, support vector machine (SVM) testing.**

I. Introduction

In data analysis, it is often useful to obtain a probability density estimate for a set of independent identically distributed (i.i.d.) observations. Such a density model can help discover underlying structures of the data in unsupervised learning, and can also yield asymptotically optimal discriminant procedures [22]. In this paper, we focus on the *finite mixture model* [30], which describes the distribution by a mixture of simple parametric functions $\phi(\cdot)$'s, as

$$f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \phi_j(\mathbf{x}). \tag{1}$$

Here, $\phi_j(\mathbf{x})$ is the jth component, α_j's are the mixing coefficients such that $\sum_{j=1}^{n} \alpha_j = 1$. The most common parametric form of ϕ is the Gaussian, leading to the well-known *Gaussian mixtures*. Finite mixture models have proven powerful in modeling complex, non-Gaussian distributions [34], [28]. It is widely used in classification, clustering, and density estimation [29], and in applications such as speech processing [33]. Typically, the model parameters are estimated by the expectation–maximization (EM) algorithm [11].

In many situations, instead of obtaining the mixture model from scratch, we are more interested in simplifying a given mixture model. The resultant, more compact model can then be used efficiently in the learning process or the subsequent testing phase. Consider the Parzen window density estimator [32]

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_j). \tag{2}$$

Here, $\{\mathbf{x}_j\}_{j=1}^{n}$ is a set of i.i.d. observations, $\mathbf{H}$ is a symmetric positive-definite bandwidth matrix $K_{\mathbf{H}}(\mathbf{x}) = K(\mathbf{H}^{-1/2}\mathbf{x})$, and $K(\cdot)$ is a multivariate kernel. This can be viewed as a special form of the mixture model, where each sample is associated with one component with uniform weighting. When n is large, this estimator becomes computationally expensive and a simplified estimator is more desirable. Another example is the decision function of the support vector machine (SVM) [36] $f(\mathbf{x}) = \sum_{j=1}^{S_v} \alpha_j K(\mathbf{x}, \mathbf{x}_j) + b$, where S_v is the number of support vectors and b is the bias term. In case K is the Gaussian kernel, the decision function is very similar in form to the Gaussian mixture model. How to reduce a nonsparse SVM decision function (i.e., the number of support vectors S_v is large) for efficient testing is an important research topic in kernel methods [3]. Other examples that involve mixture models include the particle filter [20] and nonparametric belief propagation [39], [38], where the mixture model has to be frequently used or updated during the learning process. In these circumstances, a compact model can greatly reduce the computational complexities.

One may argue that it might be easier to simply train a smaller model from scratch. However, as discussed in [18], this may not be feasible in many practical situations. For example, the original data set may be too large and may not have been kept. Moreover, the given mixture model may not be obtained directly from clustering but as a result of some other learning procedures. For example, when the model is obtained by Parzen window density estimation, it has a fixed number of components which is equal to the number of data observations. Hence, one cannot rerun the estimation procedure with fewer components. Another example is standard SVM training, where the number of support vectors cannot be specified in advance. Besides, it can be more economical in simplifying a model than retraining the whole model.

Several directions have been pursued on simplifying a given mixture model. One is based on the observation that the mixture component (such as the Gaussian) typically decays rapidly with distance. Unnecessary component summations in the model evaluation can then be circumvented by performing a neighborhood search through spatial data structures, such as the kd-trees

Manuscript received June 25, 2009; revised January 04, 2010; accepted January 07, 2010. Date of publication February 22, 2010; date of current version April 02, 2010. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region under Grant 615209.

K. Zhang is with Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: kai_zhang@lbl.gov).

J. T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: jamesk@cse.ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2040835

[25], [31]. However, such algorithms may scale poorly with the input dimensionality. Another approach, first introduced by [37] in the context of kernel density estimation, prebins the data into equally spaced mesh and then uses a modified kernel estimator on the binned data. As binning in a high-dimensional space is computationally expensive, most algorithms along this line [37], [21], [12] are focused on univariate data. Babich and Camps [1], and Joen and Landgrebe [23] perform preclustering of the data and then use the cluster centers as a reduced sample set. Girolami and He [17] proposed a reduced set density estimator using quadratic programming (QP) to obtain a sparse representation. However, the resultant QP problem still scales cubically with the sample size. Goldberger *et al.* [18] proposed a novel algorithm for learning a simplified representation of a Gaussian mixture based on the unscented transform [24]. However, this requires solving a costly eigenvalue problem to obtain the so-called sigma points. Recently, Goldberger and Roweis [19] and Davis and Dhillon [9] proposed another approach for grouping components in the Gaussian mixture model by minimizing a "local" Kullback–Leibler (KL)-based distance defined between the original and reduced mixtures. The algorithm is very efficient, and has been successfully applied in various problems such as hierarchical clustering of scenery images and handwritten digits [19], sensor networks, and statistical debugging [9] with encouraging performance.

In this paper, we propose to simplify the mixture model by minimizing an upper bound of the approximation error between the original, and the simplified model, under the use of the L_2 distance measure. This is achieved by first grouping similar components together and then performing local fitting through function approximation. The simplified model obtained can then be used as a replacement of the original model to speed up various algorithms involving mixture models during training and testing. We applied the proposed algorithm in a number of machine learning problems such as clustering and SVM testing, and obtain encouraging results.

The rest of this paper is organized as follows. Section II describes the proposed algorithm and analyzes the relationship between the two underlying steps. Section III provides comparisons with related approaches, including methods that cluster components in a mixture model and also the mean shift algorithm. In Section IV, we discuss the application of the proposed approach in simplifying the SVM decision function. The performance of the proposed method is evaluated through a number of experiments in Section V, and the last section gives some concluding remarks. Preliminary results have been reported in the conference paper [43].

II. Simplifying Mixture Models

Given a mixture model $f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \phi_j(\mathbf{x})$ (1) with a large number of components (n), we are interested in approximating it by a simplified mixture model

$$g(\mathbf{x}) = \sum_{i=1}^{m} w_i g_i(\mathbf{x}) \tag{3}$$

where $m \ll n$. Here, $\alpha_j, w_i \in \mathbb{R}$, and $\phi_j, g_i \in L_2(\mathbb{R}^d)$ are differentiable parametric functions (details will be discussed in Section II-B). Given a distance measure $D(\cdot,\cdot)$ defined on functions, the error induced by approximating f with g is

$$\mathcal{E} = D(f,g) = D\left(\sum_{j=1}^{n} \alpha_j \phi_j, \sum_{i=1}^{m} w_i g_i\right). \tag{4}$$

In this paper, we adopt the commonly used L_2 distance $D(\phi, \phi') = \sqrt{\int (\phi(\mathbf{x}) - \phi'(\mathbf{x}))^2 d\mathbf{x}}$. Since $D(\cdot,\cdot)$ is always nonnegative, it will be equivalent (but more convenient) to consider minimizing the squared distance $D^2(\cdot,\cdot)$ instead of $D(\cdot,\cdot)$ in the sequel.

A. Approximation Procedure

Since both f and g are composed of multiple components, a direct minimization of $\mathcal{E}$ in (4) is difficult. The problem can be simplified by minimizing an upper bound of $\mathcal{E}$ under the use of L_2 distance function. This allows us to decompose the optimization into easier and independent subproblems. Suppose that the (indices of) the mixture components $\{\phi_j\}_{j=1}^{n}$ are divided into disjoint clusters $\{S_1, S_2, \ldots, S_m\}$. By using the Cauchy–Schwartz inequality

$$\left(\sum_{i=1}^{m} a_i\right)^2 \leq m \sum_{i=1}^{m} a_i^2 \tag{5}$$

the model simplification error $\mathcal{E}$ (4) can be upper bounded by

$$\begin{aligned} \mathcal{E} &= \int \left(\sum_{i=1}^{m} w_i g_i(\mathbf{x}) - \sum_{j=1}^{n} \alpha_j \phi_j(\mathbf{x})\right)^2 d\mathbf{x} \\ &= \int \left(\sum_{i=1}^{m} \left(w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x})\right)\right)^2 d\mathbf{x} \\ &\leq m \sum_{i=1}^{m} \underbrace{\int \left(w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x})\right)^2 d\mathbf{x}}_{=:\bar{\mathcal{E}}_i} \\ &=: \bar{\mathcal{E}}. \end{aligned} \tag{6}$$

Note that this holds for any simplification g of f, and any partitioning $\{S_1, S_2, \ldots, S_m\}$ of $\{\phi_j\}_{j=1}^{n}$. Moreover, we can see that the upper bound $\bar{\mathcal{E}}$ comprises the local model-simplification errors $\bar{\mathcal{E}}_i$'s. Given a fixed partitioning of the mixture components, minimizing $\bar{\mathcal{E}}$ can thus be reformulated as the easier problem of minimizing the $\bar{\mathcal{E}}_i$'s (independent of each other).

A natural procedure consists of two steps. First, a good representative, $w_i g_i$ in (6), is used to simplify the ith group of ϕ_j's (*local approximation*). For clarity, let the solution be $\tilde{\mathcal{R}}_i(\mathbf{x}) = w_i g_i(\mathbf{x})$ for clarity. Then, we regroup the mixture components into compact clusters. With this new partition we go back to the local approximation step and find new $\tilde{\mathcal{R}}_i$'s which permits once more updating of the S_i's. The procedure terminates when no significant gains can be obtained.

Step 1: Local Approximation: With a given partitioning $\{S_1, S_2, \ldots, S_m\}$ of the mixture components, we minimize the local model-simplification error $\bar{\mathcal{E}}_i$ for each cluster $i (i = 1, 2, \ldots, m)$ w.r.t. w_i and $g_i(\cdot)$, i.e.,

$$\min_{w_i, g_i(\cdot)} \int \left(w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x}. \quad (7)$$

As will be discussed in Section II-B1, its solution $\tilde{\mathcal{R}}_i(\mathbf{x}) = w_i g_i(\mathbf{x})$ can be obtained via a combination of explicit formulas from differential calculus, and alternating between the updates of w_i and the parameters in the parametric form of g_i.

Step 2: Component Grouping: In this step we improve the partition. The basic idea is to group similar components into the same cluster. We measure the compactness of the ith cluster with the *local quantization error*

$$\mathcal{Q}_i = \sum_{j \in S_i} \alpha_j \int (\mathcal{R}_i(\mathbf{x}) - \phi_j(\mathbf{x}))^2 d\mathbf{x} \quad (8)$$

where $\mathcal{R}_i(\mathbf{x})$ is the cluster representative which has similar functional forms as $w_i g_i$ or $\alpha_i \phi_i$. With $\mathcal{R}_i$'s available for $i = 1, 2, \ldots, m$, we can reassign each ϕ_j $(j = 1, 2, \ldots, n)$ to its closest[1] representative $\mathcal{R}_i$ w.r.t. the distance

$$D(\phi_j, \mathcal{R}_i) = \sqrt{\int (\mathcal{R}_i(\mathbf{x}) - \phi_j(\mathbf{x}))^2 d\mathbf{x}}. \quad (9)$$

This is close in flavor to vector quantization [15] by doing which we actually find a local minimum of the distortion error

$$Q = \sum_{i=1}^{m} \sum_{j \in S_i} \alpha_j D^2(\phi_j, \mathcal{R}_i). \quad (10)$$

We have the following interesting fact that the optimal solutions in the above two steps are closely related.

Proposition 1: For each $i = 1, \ldots, m$, if $\tilde{R}_i$ is the optimal solution in the local approximation step and $\mathcal{R}_i$ the optimal solution in the component grouping step, then

$$\mathcal{R}_i(\mathbf{x}) = \tilde{\mathcal{R}}_i(\mathbf{x}) / Z_i \quad (11)$$

where $Z_i = \sum_{j \in S_i} \alpha_j$.

Proof is in Appendix I. Proposition 1 reveals a close connection between the component clustering and local approximation steps, namely that their representatives obtained only differ by a data-dependent scalar under the same partitioning of mixture components. Therefore, each iteration in the component clustering step is actually a "renormalized" version of the local approximation step. In other words, the two steps are interwoven with each other and as a result the two tasks can be achieved simultaneously. On the other hand, note that through component grouping, similar components are assigned to the same cluster. As a result, the local approximation in each cluster becomes easier and $\int (\{w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x})) d\mathbf{x}\}_{i=1,2,\ldots,m}$ in (6) are expected to be small and similar to each other. Recall that the Cauchy–Schwartz inequality (5) becomes an equality when all a_i's are the same. From this aspect, component grouping actually helps to maintain a tight upper bound of the model simplification error, which is quite beneficial to the local approximation step.

[1]As is common in clustering algorithms, we assume that the closest representative is unique.

More detailed discussions of the two steps will be presented in Section II-B. The complete algorithm works by commencing with an initial partition and alternatively performing local approximation, component grouping, local approximation, etc., until no significant further gains can be obtained.

B. Detailed Procedure

In this section, we first show in Section II-B1 how to solve the optimization problem (7) and obtain the local representative $w_i g_i(\mathbf{x})$. The complete algorithm is then discussed in Section II-B2.

We assume that the jth component of f takes the form

$$\phi_j(\mathbf{x}) = C_{\mathrm{Kd}}^{-1} |\mathbf{H}_j|^{-\frac{1}{2}} K_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j) \quad (12)$$

where $K_{\mathbf{H}_j}$ is a nonnegative, radially symmetric kernel with center $\mathbf{x}_j$, and $\mathbf{H}_j$ is a positive-definite covariance matrix that controls the bandwidth of the kernel. Usually, each $K_{\mathbf{H}}$ is bounded and satisfies the conditions

$$\int_{\mathbb{R}^d} C_{Kd}^{-1} K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} = 1 \qquad \lim_{\|\mathbf{x}\| \to \infty} \|\mathbf{x}\|^d K_{\mathbf{H}}(\mathbf{x}) = 0$$
$$\int_{\mathbb{R}^d} \mathbf{x} K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} = 0 \qquad \int \mathbf{x}\mathbf{x}^\top K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} = C_{Kd} \mathbf{I}$$

where $C_{Kd} = \int K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x}$ is a normalization factor that depends on the kernel $K_{\mathbf{H}}$ and input dimensionality d. Note that for radially symmetric kernels, a notationally simpler way to represent the kernel function is $K_{\mathbf{H}}(\mathbf{x}) = k(\mathbf{x}^\top \mathbf{H}^{-1} \mathbf{x})$, where k is called the *profile* of the kernel and takes the quadratic term $r = \mathbf{x}^\top \mathbf{H}^{-1} \mathbf{x}$ as input. For the Gaussian kernel, $k(r) = \exp(-(1/2)r)$; for the Epanechnikov kernel, $k(r) = (3/4)(1 - r)$ if $r < 1$, and $k(r) = 0$ if $r \geq 1$; whereas for the negative exponential kernel, $k(r) = \exp(-\beta r^{(1/2)})$. Similarly, each component g_i is of the form

$$g_i(\mathbf{x}) = C_{Kd}^{-1} |\tilde{\mathbf{H}}_i|^{-\frac{1}{2}} K_{\tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{t}_i) \quad (13)$$

and is associated with weight w_i, center $\mathbf{t}_i$, and covariance matrix $\tilde{\mathbf{H}}_i$. In case g is required to be normalized, w_i's should sum to 1 and this can be achieved by rescaling w_i's with $1/\sum_{i=1}^{m} w_i$.

Note that the specific form of the kernel (or, equivalently, its profile) has to be determined before a concrete algorithm can be designed. In the following, we consider the Gaussian kernel. In this case, it is easy to verify that $K_{\mathbf{H}_1}(\mathbf{x}_1) \cdot K_{\mathbf{H}_2}(\mathbf{x}_2) = \exp(-(1/2)\mathbf{x}_1^\top \mathbf{H}_1 \mathbf{x}_1) \exp(-(1/2)\mathbf{x}_2^\top \mathbf{H}_2^{-1} \mathbf{x}_2) = \exp(-(1/2)(\mathbf{x}_1^\top \mathbf{H}_1 \mathbf{x}_1 + \mathbf{x}_2^\top \mathbf{H}_2^{-1} \mathbf{x}_2))$, i.e.,

$$k(r_1) k(r_2) = k(r_1 + r_2). \quad (14)$$

This also holds for other kernels such as the negative exponential kernel. Derivations for other types of kernels will be different but Proposition 1 still holds in general.

1) Solving the Local Approximation Subproblem by Coordinate Descent: In this section, we show how to obtain the weighted function $w_i g_i(\mathbf{x})$ (with parameters $w_i, \tilde{\mathbf{H}}_i$ and $\mathbf{t}_i$) by coordinate descent. First, as is shown in Appendix II, $\bar{\mathcal{E}}_i$ in (6) can be written as

$$\bar{\mathcal{E}}_i = w_i^2 \frac{C_{Kd}^{-1}}{|2\tilde{\mathbf{H}}_i|^{\frac{1}{2}}} - w_i \sum_{j \in S_i} \frac{2C_{Kd}^{-1}\alpha_j k(r_{ij})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} + c_i \tag{15}$$

where

$$c_i = \int \left(\sum_{j \in S_i} \alpha_j C_{Kd}^{-1} |\mathbf{H}_j|^{-\frac{1}{2}} k_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j) \right)^2 d\mathbf{x}$$

is a constant irrelevant to the unknown variables ($w_i, \mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$), and

$$r_{ij} = (\mathbf{t}_i - \mathbf{x}_j)^\top (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} (\mathbf{t}_i - \mathbf{x}_j) \tag{16}$$

is the squared Mahalanobis distance (normalized by $\mathbf{H}_j + \tilde{\mathbf{H}}_i$) between the original component center $\mathbf{x}_j$ and the new component center $\mathbf{t}_i$. To minimize $\bar{\mathcal{E}}_i$ w.r.t. the unknown variables ($w_i, \mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$), one can set the corresponding partial derivatives of $\bar{\mathcal{E}}_i$ to zero. However, this leads to a nonlinear system which is difficult to solve.

Here, we decouple the relations among these three parameters and perform coordinate descent. First, recall that the quadratic function $ax^2 + bx + c$, with $a > 0$, has a unique minimum at $x = -(2a)/b$ and a minimum value of $-(b^2)/(4a) + c$. Hence, for $\bar{\mathcal{E}}_i$ which is quadratic in w_i, we choose

$$w_i = |2\tilde{\mathbf{H}}_i|^{\frac{1}{2}} \sum_{j \in S_i} \frac{\alpha_j k(r_{ij})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}}. \tag{17}$$

One can observe that (17) leads to nonnegative w_i's in the case of nonnegative α_j. Therefore, if the given model (1) represents a probabilistic model (with positive α_i's), then the resultant solution g can also be interpreted as a probabilistic model. Substituting (17) back into (15), the minimum of $\bar{\mathcal{E}}_i$ (for fixed values of $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$) is obtained as

$$\bar{\mathcal{E}}_i^{\min} = -|\tilde{\mathbf{H}}_i|^{\frac{1}{2}} \left(\sum_{j \in S_i} \alpha_j k(r_{ij}) \left| \mathbf{H}_j + \tilde{\mathbf{H}}_i \right|^{-\frac{1}{2}} \right)^2 C_{Kd}^{-1} 2^{\frac{d}{2}} + c_i. \tag{18}$$

The remaining task is to minimize $\bar{\mathcal{E}}_i^{\min}$ w.r.t. $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$. For this we need a number of matrix calculus formulas that may not be well known. Some rules that are useful in our derivations are given in Appendix IV, whose study at this point will facilitate comprehension of the details.

On setting $\partial_{\mathbf{t}_i} \bar{\mathcal{E}}_i^{\min} = 0$, we have

$$\mathbf{t}_i = \mathbf{M}_i^{-1} \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}) (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} \mathbf{x}_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} \tag{19}$$

where

$$\mathbf{M}_i = \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}) (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}}.$$

For a fixed $\tilde{\mathbf{H}}_i$, we can obtain $\mathbf{t}_i$ by starting with an initial $\mathbf{t}_i^{(0)}$, and then iterate (19) (recall that r_{ij} is a function of $\mathbf{t}_i$). As will be discussed in Section III-B, (19) is identical to the *variable bandwidth mean shift* procedure [4]. When $\mathbf{H}_j$'s are spherical, the local convergence of this iteration has been shown in [4] and [8].

As for $\tilde{\mathbf{H}}_i$, we set $\partial_{\tilde{\mathbf{H}}_i} \bar{\mathcal{E}}_i^{\min} = 0$ and obtain

$$\begin{aligned} &\sum_{j \in S_i} \frac{\alpha_j k(r_{ij})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} \left(\frac{1}{2} \tilde{\mathbf{H}}_i^{-1} - (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} \right) \\ &\quad - 2 \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} \cdot (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} \\ &\quad \times (\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)^\top (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} = 0. \end{aligned}$$

Substitute the right-hand side of $(1/2)\tilde{\mathbf{H}}_i^{-1} - (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} = (1/2)(\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1}(\mathbf{H}_j - \tilde{\mathbf{H}}_i)\tilde{\mathbf{H}}_i^{-1}$ in the above, and multiply it on the right with $\tilde{\mathbf{H}}_i$. The resultant formula will then have in its left term factors $(\mathbf{H}_j - \tilde{\mathbf{H}}_i)$. One can now isolate $\tilde{\mathbf{H}}_i$ and obtain

$$\begin{aligned} \tilde{\mathbf{H}}_i = \mathbf{P}_i^{-1} \sum_{j \in S_i} \frac{\alpha_j (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} &(k(r_{ij}) \mathbf{H}_j \\ -4k'(r_{ij})(\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)^\top &(\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1} \tilde{\mathbf{H}}_i) \end{aligned} \tag{20}$$

where

$$\mathbf{P}_i = \sum_{j \in S_i} \frac{(\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{\frac{1}{2}}} \alpha_j k(r_{ij}).$$

Note that in the special case of spherical $\mathbf{H}_j$'s and $\tilde{\mathbf{H}}_i$'s, $\tilde{\mathbf{H}}_i$ can be simply obtained as a closed-form solution and so iteration (20) is not needed. Details will be presented in Section IV.

Given some initial choices of $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$, we thus repeatedly update (19) and (20) until both $\|\mathbf{t}_i^{(t+1)} - \mathbf{t}_i^{(t)}\|_2$ and $\|\tilde{\mathbf{H}}_i^{(t+1)} - \tilde{\mathbf{H}}_i^{(t)}\|_F$ are close to zero (here, $\|\cdot\|_F$ denotes the Frobenius norm). After obtaining $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$, they are substituted into (17) to compute w_i. For initialization, we follow [19] and set

$$\begin{aligned} \mathbf{t}_i^{(0)} &= \frac{\sum_{j \in S_i} \alpha_j \mathbf{x}_j}{\sum_{j \in S_i} \alpha_j} \\ \tilde{\mathbf{H}}_i^{(0)} &= \frac{1}{\sum_{j \in S_i} \alpha_j} \sum_{j \in S_i} \alpha_j \left(\mathbf{H}_j + \left(\mathbf{t}_i^{(0)} - \mathbf{x}_j \right) \left(\mathbf{t}_i^{(0)} - \mathbf{x}_j \right)^\top \right) \end{aligned} \tag{21}$$

which are the weighted mean and covariance of the samples, respectively. Both can be computed efficiently. It is worthwhile to note that this initialization can be regarded mathematically as moment matching, i.e., requiring (the normalized versions of) the local component group $\sum_{j \in S_i} \alpha_j \phi_j$ and the reduced model $w_i g_i(\mathbf{x})$ in (13) to have the same mean and covariance. In [44], a simpler scheme is applied in simplifying a group of spherical Gaussian components into a single one, by using the same bandwidth parameter as those of the original components. Asymptotic behavior of such density estimator was analyzed in [44].

2) The Complete Algorithm: With the solution $\tilde{\mathcal{R}}_i = w_i g_i$ in the local approximation step, the representative $\mathcal{R}_i$ in the component clustering step can also be easily obtained by the connection stated in Proposition 1. The whole algorithm is presented in Algorithm 1. The outer loop (lines 1–28) is indexed by t. Inside this loop, we process each of the m clusters $S_1, S_2, \ldots, S_m$.

Algorithm 1. Algorithm for simplifying mixture models. Notations: $r_{ij}^{p,q} = (\mathbf{t}_i^{(p)} - \mathbf{x}_j)^\top (\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)})^{-1} (\mathbf{t}_i^{(p)} - \mathbf{x}_j)$, $i, j, p, q \in \mathbb{Z}$, $\mathbf{t}^{(p)}, \mathbf{x}_j \in \mathbb{R}^d$, $\mathbf{H}_j, \tilde{\mathbf{H}}_i^{(q)} \in \mathbb{R}^{d \times d}$; Gaussian kernel $k(r) = \exp(-\frac{1}{2}r)$ and $k'(r) = -\frac{1}{2}\exp(-\frac{1}{2}r)$. The derivation of $D(\phi_j, \mathcal{R}_i)$ in line 23 can be found in Appendix C.

```
Input: ${\alpha_j, \phi_j(\mathbf{x}_j, \mathbf{H}_j)}_{j=1}^n$; initial partitioning of $\phi_j$'s into ${S_1, S_2, \ldots, S_m}$;
$t = 0$, $e^{(t)} = 10^{10}$; $\epsilon_1 = 0.01, \epsilon_2 = 0.01$.
Output: $\left\{w_i, g_i\left(\mathbf{t}_i, \tilde{\mathbf{H}}_i\right)\right\}_{i=1}^m$.
1  repeat
2      $t = t + 1$;
3      for $i = 1, 2, \ldots, m$ do
4          $\mathbf{t}_i^{(0)} = \frac{\sum_{j \in S_i} \alpha_j \mathbf{x}_j}{\sum_{j \in S_i} \alpha_j}$;
5          $\tilde{\mathbf{H}}_i^{(0)} = \frac{1}{\sum_{j \in S_i} \alpha_j} \sum_{j \in S_i} \alpha_j \left(\mathbf{H}_j + (\mathbf{t}_i^{(0)} - \mathbf{x}_j)(\mathbf{t}_i^{(0)} - \mathbf{x}_j)^\top\right)$;
6          repeat
7              $p = 0$;
8              repeat
9                  $\mathbf{t}_i^{(p+1)} = \left(\sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}^{p,0})(\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)})^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)}|^{\frac{1}{2}}}\right)^{-1} \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}^{p,0})(\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)})^{-1} \mathbf{x}_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)}|^{\frac{1}{2}}}$;
10                 $p = p + 1$;
11             until $\|\mathbf{t}_i^{(p)} - \mathbf{t}_i^{(p-1)}\|_F \leq \epsilon_1$ ;
12             $q = 0$;
13             repeat
14                 $\tilde{\mathbf{H}}_i^{(q+1)} = \left(\sum_{j \in S_i} \frac{(\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1} \alpha_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^{\frac{1}{2}}} k(r_{ij}^{p,q})\right)^{-1} \sum_{j \in S_i} \frac{\alpha_j (\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^{\frac{1}{2}}} \left(k(r_{ij}^{p,q})\mathbf{H}_j - 4k'(r_{ij}^{p,q})(\mathbf{x}_j - \mathbf{t}_i^{(p)})(\mathbf{x}_j - \mathbf{t}_i^{(p)})^\top (\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1} \tilde{\mathbf{H}}_i^{(q)}\right)$;
15                 $q = q + 1$;
16             until $\|\mathbf{H}_i^{(q)} - \mathbf{H}_i^{(q-1)}\|_F \leq \epsilon_2$ ;
17             $w_i = |2\tilde{\mathbf{H}}_i^{(q)}|^{\frac{1}{2}} \sum_{j \in S_i} \frac{\alpha_j k(r_{ij}^{p,q})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^{\frac{1}{2}}}$;
18         until $p \leq 1$ and $q \leq 1$ ;
19     end
20     $e^{(t)} = 0$;
21     for $j = 1, 2, \ldots, n$ do
22         for $i = 1, 2, \ldots, m$ do
23             $D(\phi_j, \mathcal{R}_i) = C_{Kd}^{-1}\left(|2\mathbf{H}_j|^{-\frac{1}{2}} + \frac{w_i^2}{\mathbf{Z}_i^2}|2\tilde{\mathbf{H}}_i|^{-\frac{1}{2}} - 2\frac{w_i}{\mathbf{Z}_i}k(r_{ij}^{p,q})|\tilde{\mathbf{H}}_i + \mathbf{H}_j|^{-\frac{1}{2}}\right)$;
24         end
25         Assign $\phi_j$ to cluster $S_i$ such that $i = \arg\min_k D(\phi_j, \mathcal{R}_k)$;
26         $e^{(t)} = e^{(t)} + D(\phi_j, \mathcal{R}_i)$;
27     end
28 until $|e^{(t)} - e^{(t-1)}| \leq 0.001 |e^{(t)}|$ ;
29 Obtain the simplified model as $g(\mathbf{x}) = \sum_{i=1}^m Z_i \mathcal{R}_i(\mathbf{x}) = \sum_{i=1}^m w_i C_{Kd}^{-1} |\tilde{\mathbf{H}}_i|^{-\frac{1}{2}} K_{\tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{t}_i)$, where $Z_i = \sum_{j \in S_i} \alpha_j$.
```

For each cluster, we first initialize $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$ (lines 4–5), and then alternate between the updating of $\mathbf{t}_i$ (lines 8–11, p steps) and the updating of $\tilde{\mathbf{H}}_i$ (lines 13–16, q steps). When both p and q are smaller than a small integer (1 in our algorithm), $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$ have converged. Then, we compute the distances between each component ϕ_j and the representatives $\mathcal{R}_i$'s (lines 20–27), update the partitioning by assigning ϕ_j to the closest representative $\mathcal{R}_i$, and start the next iteration.

C. Time Complexity

Recall that n is the number of components in the original mixture, m is the number of clusters, and d is the data dimensionality. Let n_i be the size of cluster i, T be the number of steps for the outermost loop, and S be the number of iterations inside the loop in lines 6–19. So the overall complexity is $T(S\sum_{i=1}^m n_i d^3(p+q) + nmd^3) = Td^3 n(S(p+q)+m)$.

For computational efficiency, we can use diagonal or even spherical bandwidth matrices. If the covariances of the mixture components are also diagonal (as in kernel density estimators, SVM testing, or belief propagation), then the d^3 term in the complexity is reduced to linear, and q becomes 1 [because then there exists a closed-form solution for the iteration in lines 13–16 as will be shown in (29)]. In practice, it is observed that the number of iterations S, T, p, q is typically small. In particular, we find that T is around 15; and S does not quite affect the performance. Therefore, in practice, we set S to a very small number, i.e., we only perform a few alternations between updating $\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$ inside each of the outer loops.

On the other hand, the algorithm in [19] and [9] has a complexity of $O(lmnd^3)$ (l is the number of iterations) when g_i's have full covariances, and $O(lmnd)$ when both g_i's and mixture components have diagonal covariances. Empirically, it is more efficient than our approach since the mean/covariance of each component can be conveniently computed in one single step as the weighted mean/covariance of the local cluster of components. The comparison of the time consumptions can be found in Section V-C.

III. Comparison With Related Methods

A. Component-Clustering Methods for Mixture Models

There have been several related works on grouping the components in a mixture model. For example, Banerjee *et al.* [2] propose a general clustering framework based on the Bregman divergence, and find a local minimum of the Bregman information similar to the quantization error in (10). Most related to ours is the work of Goldberger and Roweis [19] that uses an iterative scheme to cluster multivariate Gaussian functions. There, a Gaussian mixture f in (1) is approximated by a simpler mixture g in (3) by minimizing the following "local KL-distance"

$$d(f,g) = \sum_{j=1}^{n} \alpha_j \mathrm{KL}(\phi_j \| g_{\pi(j)}) \tag{22}$$

where $\mathrm{KL}(\cdot\|\cdot)$ is the KL-divergence, and $\pi(j)$ maps component ϕ_j to its closest component $g_{\pi(j)}$ [i.e., $\pi(j) = \arg\min_{i=1,2,\ldots,m} \mathrm{KL}(\phi_j\|g_i)$]. To minimize (22), an iterative procedure is applied which alternates between finding the partition and reassigning the components to the closest representatives. At each iteration, the parameters ($\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$) of g_i are determined as in (21). Davis and Dhillon [9] also derived the same procedure by expressing the differential entropy between two multivariate Gaussians as a convex combination of the Mahalanobis distance between the Gaussian means and the Burg matrix divergence between the covariance matrices. As can be seen, Goldberger and Roweis [19] and Davis and Dhillon [9] consider the problem of *component clustering*, i.e., grouping a set of Gaussian components ϕ_i's, based on minimizing the distortion measure in (22). On the other hand, our algorithm is motivated from minimizing (an upper bound of) the model simplification error, and the resultant simplified model can be used as a replacement of the original mixture model. Our goal is realized through the two steps of component grouping and local approximation, and in particular, the component grouping step helps maintain the tightness of the bound so that our optimization can produce desired result. Moreover, note that Goldberger and Roweis [19] and Davis and Dhillon [9] adopt the KL-divergence in measuring the distance between Gaussian components, while we use the L_2 distance to measure the distance between two mixtures.

There are also interesting differences between the proposed method and [19] and [9] in the choice of the local covariance $\tilde{\mathbf{H}}_i$. This can be seen more clearly when we consider the special case where the given mixture model f is a Parzen window density estimator (2). In other words, all the ϕ_j's have the same weight and covariance matrix ($\mathbf{H}_j = \mathbf{H}$ for $j = 1, 2, \ldots, n$, where $\mathbf{H}$ is the bandwidth of the original components). Bandwidth (20) of the representative kernel g_i can then be rewritten as

$$\tilde{\mathbf{H}}_i = \mathbf{H} + 4\tilde{\mathbf{H}}_i(\tilde{\mathbf{H}}_i + \mathbf{H})^{-1}\mathbf{V}_i$$

where

$$\mathbf{V}_i = \frac{\sum_{j\in S_i} -\alpha_j k'(r_{ij})(\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)^\top}{\sum_{j\in S_i} \alpha_j k(r_{ij})}$$

is a weighted covariance of the local cluster S_i, with $-\alpha_j k'(r_{ij})/\sum_{j\in S_i}\alpha_j k(r_{ij})$ being the weight on sample $\mathbf{x}_j$. Recall that r_{ij} is the squared Mahalanobis distance between $\mathbf{x}_j$ and center $\mathbf{t}_i$ normalized by $(\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1}$. For kernels such as Gaussian and Laplacian, $-k'(r_{ij})$ decreases with r_{ij}. Thus, $\mathbf{V}_i$ can be regarded as a robust covariance estimate that assigns smaller weights to points far away from the center $\mathbf{t}_i$. Interestingly, this distance-based weighting is similar to that in manifold Parzen windows [40], which is designed to handle sparse, high-dimensional data in a more robust manner. In contrast, Goldberger and Roweis [19] and Davis and Dhillon [9] choose $\tilde{\mathbf{H}}_i = \mathbf{H} + \mathrm{Cov}[S_i]$, where $\mathrm{Cov}[S_i]$ is simply the (nonrobust) covariance of S_i.

B. Mean Shift Algorithm

Mean shift [6], [13] is an iterative mode-seeking algorithm widely used in pattern recognition and computer vision. It assumes that the local maxima of the underlying density distribution correspond to clusters of the data. By shifting every data point iteratively along the gradient of the density, dominant clusters can be identified. Specifically, given a sample set $\{x_i\}_{i=1}^n$, the Parzen window density estimator (2) is

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} C_{Kd}^{-1}|\mathbf{H}|^{-\frac{1}{2}} K_{\mathbf{H}}(\mathbf{x}-\mathbf{x}_i) \tag{23}$$

where $K_{\mathbf{H}}(\mathbf{x}) = k(\mathbf{x}^\top\mathbf{H}^{-1}\mathbf{x})$ is the kernel with prespecified covariance $\mathbf{H}$. By setting $\nabla\hat{f}(\mathbf{x}) = 0$, we have

$$\frac{1}{n}\sum_{i=1}^{n} C_{Kd}^{-1}|\mathbf{H}|^{-\frac{1}{2}} k'((\mathbf{x}-\mathbf{x}_i)'\mathbf{H}^{-1}(\mathbf{x}-\mathbf{x}_i))\cdot 2\mathbf{H}^{-1}(\mathbf{x}-\mathbf{x}_i) = 0.$$

After some reductions, this can be written in iterative form [6]

$$\mathbf{x}^{(t+1)} \longleftarrow \frac{\sum_{i=1}^{n} k'((\mathbf{x}^{(t)}-\mathbf{x}_i)^\top\mathbf{H}^{-1}(\mathbf{x}^{(t)}-\mathbf{x}_i))\mathbf{x}_i}{\sum_{i=1}^{n} k'((\mathbf{x}^{(t)}-\mathbf{x}_i)^\top\mathbf{H}^{-1}(\mathbf{x}^{(t)}-\mathbf{x}_i))}. \tag{24}$$

It has been shown [6] that iteration (24) is convergent and can find the local maxima of the density (23). Clustering is then obtained by associating each data sample with its corresponding density maximum. The mean shift algorithm does not make prior assumptions on the shape of the clusters, and the number of classes can be determined automatically. It has demonstrated great success in many vision applications, such as image segmentation [5], [6], tracking [7], [20], [45], and video processing [10], [26]. Recently, Zhang *et al.* introduced kernel smoothing into the complex network analysis and successfully applied the mean shift algorithm for multiresolution community detection [41].

Considering the special case of fitting Parzen window density estimator as in Section III-A, the iteration (19) for determining center $\mathbf{t}_i$ of the representative g_i is reduced to

$$\mathbf{t}_i = \frac{\sum_{j\in S_i} k'\left((\mathbf{x}_j - \mathbf{t}_i)^\top(\mathbf{H}+\tilde{\mathbf{H}}_i)^{-1}(\mathbf{x}-\mathbf{x}_j)\right)\mathbf{x}_j}{\sum_{j\in S_i} k'\left((\mathbf{x}_j - \mathbf{t}_i)^\top(\mathbf{H}+\tilde{\mathbf{H}}_i)^{-1}(\mathbf{x}-\mathbf{x}_j)\right)}. \quad (25)$$

Comparing this with (24), it is clear that (25) is actually a mean-shift procedure that locates the peak of a new "surrogate" density function

$$p_i(\mathbf{x}) = C_{Kd}^{-1}\sum_{\mathbf{x}_j\in S_i}|\mathbf{H}+\tilde{\mathbf{H}}_i|^{-1/2}K_{\mathbf{H}+\tilde{\mathbf{H}}_i}(\mathbf{x}-\mathbf{x}_j). \quad (26)$$

Note that when the covariance matrices $\mathbf{H}_j$'s associated with each component are different, the iteration (25) for finding the center $\mathbf{t}_i$ will be similar to the so-called *variable bandwidth mean shift* [4], [8].

IV. Application to SVM Testing

In this section, we consider how to speed up SVM testing by applying the proposed model simplification scheme. We assume the use of the Gaussian kernel $K(\mathbf{x},\mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2h^2)$. The SVM decision function can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{n}\beta_i y_i \exp(-\|\mathbf{x}-\mathbf{x}_i\|^2/2h^2) + b \quad (27)$$

where β_i's are the Lagrange multipliers, and $\mathbf{x}_i$'s are the support vectors with corresponding labels y_i's. Equation (27) is similar in form to the Gaussian mixture model in (1), where all components share the same (spherical) covariance $\mathbf{H}_i = h_i^2\mathbf{I}$ ($\mathbf{I}$ is the $d\times d$ identity matrix), with weight $\alpha_i = y_i\beta_i$. Care should be taken in that 1) the component weights $\alpha_i y_i$'s in (27) can be positive and negative; and 2) the components are unnormalized. For the first issue, we separate the model into positive and negative parts, and treat them separately. For the second issue, note that the missing normalization factor $|\mathbf{H}|^{-1/2} = h^{-d}$ is a constant. Therefore, all the conclusions in Section II-B apply here, except that a rescaling constant h^d (i.e., the inverse of the missing normalization factor) is needed when computing the coefficients w_i's of the components g_i's.

Fitting a mixture model using full covariance matrices introduces d^2 variables for each component, which can greatly slow down the kernel evaluation on high-dimensional data. So we require the covariances $\tilde{\mathbf{H}}_i$'s of the simplified mixture model in (3) to be "spherical", i.e., $\tilde{\mathbf{H}}_i = \tilde{h}_i^2\mathbf{I}$. Plugging it into (18), we obtain the local model-simplification error as

$$\bar{\mathcal{E}}_i^{\min} = \left(\frac{\tilde{h}_i}{h^2+\tilde{h}_i^2}\right)^d\left(\sum_{j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2\left(h^2+\tilde{h}_i^2\right)}\right)\right)^2.$$

By using a similar procedure as in Section II-B, we obtain the following analytic solution for the component center:

$$\mathbf{t}_i = \frac{\sum_{\mathbf{x}_j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2(h^2+\tilde{h}_i^2)}\right)\mathbf{x}_j}{\sum_{\mathbf{x}_j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2(h^2+\tilde{h}_i^2)}\right)}$$

which is in the form of fixed-point iteration. Similarly, the update equation for $\tilde{h}^2$ can be obtained as

$$\tilde{h}_i^2 = h^2 + \frac{2\tilde{h}_i^2}{h^2+\hat{h}_i^2}V_i \quad (28)$$

where

$$V_i = \frac{1}{d}\frac{\sum_{\mathbf{x}_j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2(h^2+\tilde{h}_i^2)}\right)\|\mathbf{t}_i-\mathbf{x}_j\|^2}{\sum_{\mathbf{x}_j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2(h^2+\tilde{h}_i^2)}\right)}.$$

Note that the iteration (28) is a simple scalar equation, which will converge to

$$\tilde{h}_i^2 = V_i + \sqrt{h^4+V_i^2}. \quad (29)$$

After fixing the component center $\mathbf{t}_i$ and covariance $\tilde{h}_i^2$, we can obtain the weight w_i for each component by using (17), as

$$w_i = h^d\left(\frac{2\tilde{h}_i^2}{h^2+\tilde{h}_i^2}\right)^{\frac{d}{2}}\sum_{\mathbf{x}_j\in S_i}\alpha_j\exp\left(-\frac{\|\mathbf{t}_i-\mathbf{x}_j\|^2}{2(h^2+\tilde{h}_i^2)}\right). \quad (30)$$

Here, we have multiplied back the inverse of the normalization factor $|\mathbf{H}|^{1/2} = h^d$, because the SVM decision function (27) contains unnormalized mixture components. Note that another choice for computing w_i, after the mean $\mathbf{t}_i$ and covariance $\tilde{h}_i^2\mathbf{I}$ of the component g_i are fixed, is to let $w_i g_i(\mathbf{x})$ and $\sum_{j\in S_i}\alpha_j\phi_j(\mathbf{x})$ have the same zeroth-order moment. This naturally leads to $w_i = \sum_{j\in S_i}\alpha_j$, and by doing this for all the clusters S_i's, $i = 1,2,\ldots,m$, the original model $f = \sum_{j=1}^n \alpha_j\phi_j$ and the simplified model $g = \sum_{i=1}^m w_i g_i$ will also have the same zeroth-order moment. In other words, if f is normalized, then g will also be normalized. In our experiments, this strategy gives similar performance as (30).

We can also design the spherical covariance version of the method by [19]. There, the local KL-divergence objective (22) can be written as

$$\begin{aligned}&\sum_{j\in S_i}\alpha_j\mathrm{KL}(\phi_j(\mathbf{x})\|g_i(\mathbf{x}))\\ &\quad=\sum_{j\in S_i}\alpha_j\left(\mathrm{tr}\left(\mathbf{H}_j\tilde{\mathbf{H}}_i^{-1}\right) - \log\left|\mathbf{H}_j\tilde{\mathbf{H}}_i^{-1}\right| - d\right.\\ &\qquad\left.+(\mathbf{x}_j-\mathbf{t}_i)'\tilde{\mathbf{H}}_i^{-1}(\mathbf{x}_j-\mathbf{t}_i)\right).\end{aligned}$$

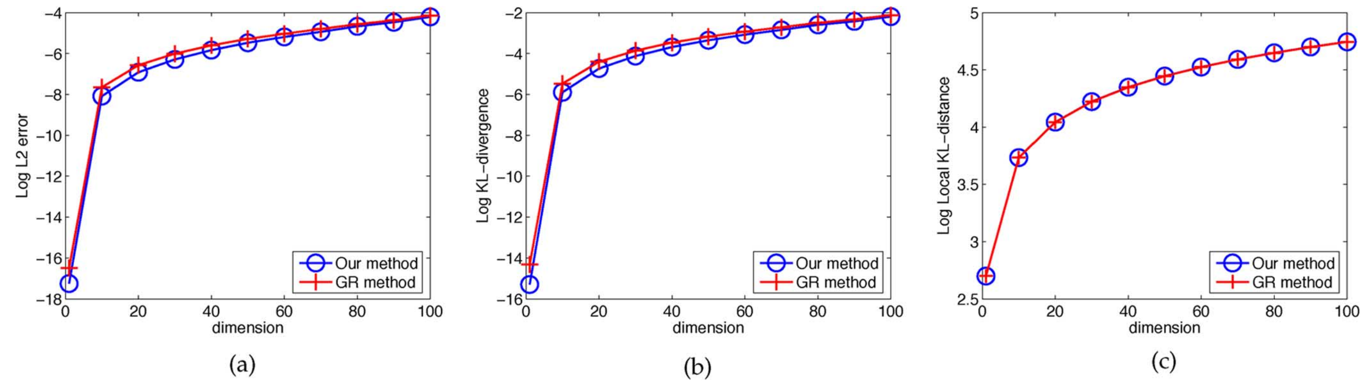

Fig. 1. Performance of model simplification with different input dimensionalities. (a) L_2 distance. (b) KL-divergence. (c) Local KL-distance.

By noting that all the $\mathbf{H}_j$'s are equal to $h^2\mathbf{I}$, this can be simplified as

$$\sum_{j\in S_i}\alpha_j \mathrm{KL}(\phi_j(\mathbf{x})\|g_i(\mathbf{x})) = \sum_{j\in S_i}\alpha_j\left(d\frac{h^2}{\tilde{h}_i^2} - 2d\log\left(\frac{h}{\tilde{h}_i}\right) - d + \frac{\|\mathbf{x}_j-\mathbf{t}_i\|^2}{\tilde{h}_i^2}\right). \quad (31)$$

Computing the derivative of (31) w.r.t. $\mathbf{t}_i$ and $\tilde{h}_i$, we obtain the component centers

$$\mathbf{t}_i = \frac{\sum_{\mathbf{x}_j\in S_i}\alpha_j\mathbf{x}_j}{\sum_{\mathbf{x}_j\in S_i}\alpha_j}$$

and the spherical covariances

$$\tilde{h}_i^2 = h^2 + \frac{1}{d}\frac{\sum_{j\in S_i}\alpha_j\|\mathbf{t}_i-\mathbf{x}_j\|^2}{\sum_{j\in S_i}\alpha_j}.$$

The weight w_i is still the summation of the α_j's based on the moment matching criteria, i.e., $w_i = h^d\sum_{j\in S_i}\alpha_j$, with the extra term h^d due to the missing normalization factor. After obtaining the parameters $\{w_i, \mathbf{t}_i, \tilde{h}_i^2\}$'s for the positive and negative parts of the SVM decision function (27), we can approximate the decision function by

$$f(\mathbf{x}) \approx \sum_{i\in\text{Positive}}\frac{w_i}{\tilde{h}_i^d}\exp\left(-\|\mathbf{x}-\mathbf{t}_i\|^2/2\tilde{h}_i^2\right) + \sum_{i\in\text{Negative}}\frac{w_i}{\tilde{h}_i^d}\exp\left(-\|\mathbf{x}-\mathbf{t}_i\|^2/2\tilde{h}_i^2\right) + b.$$

The factor C_{Kd}^{-1} is not needed here because it is not included in the decision function (27).

V. Experiments

In this section, we perform several experiments to evaluate the proposed mixture simplification scheme.[2] We assume that the original mixture is already given by the user. Issues related to how this mixture is obtained, such as model selection and parameter estimation of the original mixture, are thus beyond the scope of this paper.

We use a number of different tasks to evaluate the algorithm performance. In Section V-A, we compare our method with [19] (called "GR" in the sequel) on approximating a toy mixture model with varying data dimensionality. In Section V-B, we apply the proposed model simplification scheme to scale up mean shift, a density-based clustering algorithm widely used in vision problems such as image segmentation. We also compare our approach with the spatial-structure-based method (kd-tree) in accelerating this specific clustering algorithm. In Section V-C, we apply the two model simplification methods in speeding up SVM testing. The SVM decision function is approximated by using only a model whose number of components is only 5% of the number of support vectors.

A. *Simplifying Toy Mixture Models*

In this section, we generate a Gaussian mixture with 500 components. The data is d-dimensional, with the data value for each dimension uniformly distributed in $[0, 1]$. The weight of each Gaussian component is randomly chosen in $[0, 1]$, and its covariance h is fixed at 0.5. The number of components in the simplified model is fixed at $m = 20$. We gradually increase d from 1 to 100. The experiment is repeated 10 times and we report the average model-simplification error on a test set. The error criteria used are L_2 error, standard KL-divergence, and local KL-distance defined in (22). Note that the local KL-distance is dependent on the partitioning of the components. Therefore, we use the same partitioning for both methods in order to have a meaningful comparison.

Results are shown in Fig. 1. As can be seen, in terms of the L_2 and KL distances, our method is more accurate (on average, ours is 60.17% and 60.49% of those by GR, respectively). With the local KL-distance, the GR method is more accurate (on average, it is equal to 99.90% of ours). Note that for a fixed partitioning of the mixture components, the GR method is known to give the optimal solution under the local KL-distance measure [19]. Moreover, observe that the errors of both methods grow exponentially with dimensionality. This is because the width h of the Gaussian component has been kept constant. With increasing dimensionality, the data points become further apart and each Gaussian component gradually shrinks and ultimately becomes

[2]The Matlab codes of our method can be downloaded from http://www.cse.ust.hk/~twinsen/Simplify_GMM.zip.

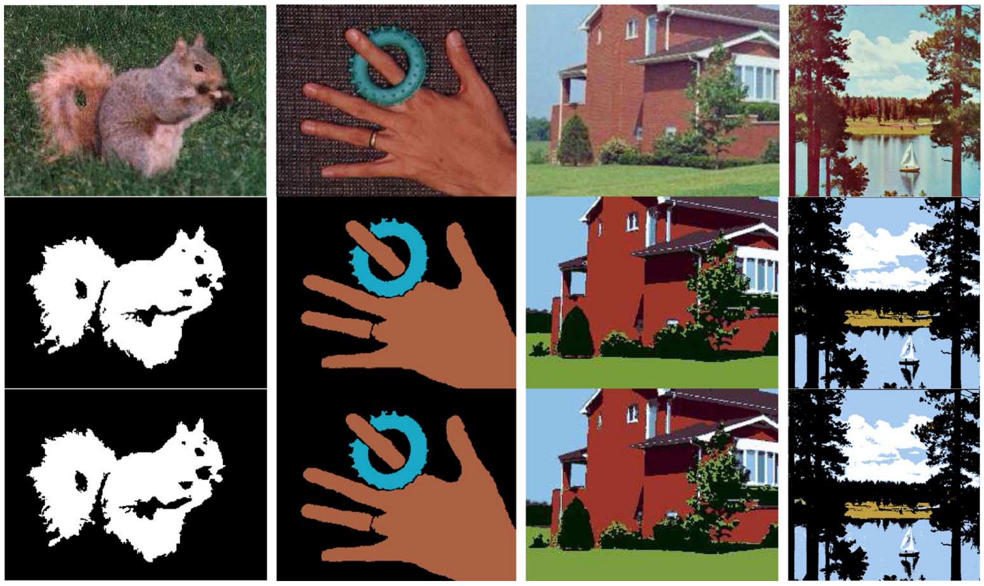
Fig. 2. Segmentation results. Row 1: Original image. Row 2: Standard mean shift. Row 3: Our method.

TABLE I
TIME CONSUMPTIONS (IN SECONDS) OF 1) STANDARD MEAN SHIFT; 2) ITS FAST VERSION USING KD-TREES (KD-MEAN-SHIFT); AND 3) ITS FAST VERSION USING OUR METHOD (OURS). THE NUMBER OF COMPONENTS (m) IN THE SIMPLIFIED MODEL IS SHOWN IN BRACKETS

image	image size (n)	standard mean shift	kd-mean-shift	ours
squirrel	209×288 (60,192)	1215.8	11.94	0.18 (81)
hand	243×302 (73,386)	1679.7	12.92	0.35 (120)
house	192×255 (48,960)	1284.5	5.16	0.22 (159)
lake	512×512 (262,144)	3343.0	85.65	3.67 (440)

an impulse-like function in high-dimensional spaces. Since the number of components is kept unchanged in the experiment, approximation becomes difficult when the dimensionality is high.

B. Clustering-Based Image Segmentation

In this section, we demonstrate the usefulness of the proposed model simplification scheme in a practical clustering algorithm, the mean shift [6]. As is briefly reviewed in Section III-B, the mean shift algorithm is a density-based, nonparametric mode-seeking algorithm that has been widely used in various computer vision problems such as tracking, segmentation, and video processing. However, the density model involved in mean shift is the Parzen window estimator whose number of components equals the sample size n. Therefore, mean shift has a time complexity of $O(n^2)$ [14] and can be expensive on large data sets. In this experiment, we start by approximating the given Parzen window density estimator $(\hat{f})$ with a compact mixture (g), and then perform clustering by detecting modes on the simplified model g. The complexity can then be reduced to $O(m^2)$, where m is the number of components in g.

We use benchmark images that have been extensively tested in the mean shift literature[3] [6] (Fig. 2). The 3-D RGB colors are used as features. Considering that the different images have varying degrees of complexity, the use of a single m for all the images might not be suitable. So we take extra care in choosing the seeds of vector quantization such that its initial partitions have roughly the same radius r (which is set to $r = 25$ in the experiments). By doing this, images with larger variations in color (or larger point clouds in the RGB space) will be approximated by more components (or a larger m). Segmentation results are shown in Fig. 2. As can be seen, our segmentation results are very close to those of the standard mean shift algorithm. The time consumption and the number of components m are shown in Table I. As expected, the number of components m is significantly smaller than the original component size n used in the Parzen window estimator.

We also compare our approach with an accelerated variant of mean shift (labeled "kd-mean-shift" in Table I), which uses the kd-tree[4] to speed up neighborhood search in computing the mean shift vector. Specifically, recall that mean shift iterations shift the point at $\mathbf{x}$ using (24). In case of the Gaussian kernel with bandwidth matrix $\mathbf{H} = h^2\mathbf{I}$, each sample $\mathbf{x}_i$ is assigned a weight $k'(-\|\mathbf{x}-\mathbf{x}_i\|^2/2h^2)$ which can be ignored if $\|\mathbf{x}-\mathbf{x}_i\|$ is larger than a threshold τ (say, $\tau = 3h$). So one only needs to search for neighbors $\mathbf{x}_i$'s of $\mathbf{x}$ inside $\|\mathbf{x}-\mathbf{x}_i\| \leq \tau$. This (spherical) range search can be done using spatial data structures like the kd-trees [35] and locality sensitive hashing [16], [14]. The kd-trees have been used in this manner for accelerating kernel summations in the EM algorithm [31]. Our method can also utilize kd-trees for additional speedup, though this is not implemented because the number (m) of centers $\mathbf{t}_i$'s in the simplified model g is already quite small (Table I). So we simply

[3]http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

[4]The code is downloaded from the ANN library (http://www.cs.umd.edu/~mount/ANN).

TABLE II
BREAKDOWN OF THE TIME CONSUMPTION (IN SECONDS) BY OUR METHOD AND THE KD-TREE-BASED MEAN SHIFT

	build model		range searching		clustering		neighborhood size		total time	
image	kd-tree	ours	kd-tree	ours	kd-tree	ours	kd-tree	ours	kd-tree	ours
squirrel	0.25	0.16	4.247	0.0073	7.44	0.013	2237	2.654	11.94	0.18
hand	0.47	0.31	2.538	0.0081	9.91	0.032	2832	3.8	12.92	0.35
house	0.20	0.15	1.366	0.0046	3.59	0.066	2090	3.352	5.16	0.22
lake	1.12	3.21	21.68	0.0341	62.85	0.421	3321	3.754	85.65	3.67

TABLE III
THE USPS DIGIT DATA SET AND THE PARAMETERS USED

digits	#training	#test	dim	γ	C	#SV (n)	m
0-3	1852	525	256	233.7	100	133	6
2-3	1389	364	256	118.7	100	283	14
2-4	1383	398	256	237.9	1000	199	9
2-8	1273	364	256	115.2	1	309	15
3-5	1214	326	256	105.8	10	353	16
3-8	1200	332	256	200.9	10	191	8
4-7	1297	347	256	92.3	10	265	12
4-9	1296	377	256	176.5	100	183	8
5-6	1220	330	256	103.9	100	168	8
5-8	1098	326	256	219.3	10	185	8
7-9	1289	324	256	149.7	100	217	10
8-9	1186	343	256	438.3	100	102	4

TABLE IV
TESTING ERRORS (IN PERCENT) ON THE USPS DIGIT SET

digits	original	GR	ours	t (confidence)
0-3	1.524	1.990±0.377	1.707±0.237	8.813(99.9%)
2-3	2.473	3.264±0.689	2.728±0.277	8.705(99.9%)
2-4	1.759	3.658±0.853	3.196±0.459	6.252(99.9%)
2-8	3.297	3.533±0.438	3.332±0.340	4.615(99.9%)
3-5	5.215	6.310±0.901	5.620±0.832	8.928(99.9%)
3-8	2.410	3.337±0.495	3.208±0.433	3.116(99.5%)
4-7	2.594	3.438±0.709	3.147±0.375	4.113(99.9%)
4-9	2.918	3.167±0.794	2.833±0.386	4.006(99.9%)
5-6	1.212	1.564±0.770	1.227±0.323	4.175(99.9%)
5-8	2.761	3.414±0.551	3.202±0.452	4.246(99.9%)
7-9	2.469	7.102±3.247	4.728±1.074	8.214(99.9%)
8-9	1.166	1.201±0.224	1.134±0.124	2.666(99.5%)

TABLE V
TIME CONSUMPTION (IN SECONDS) ON THE USPS DIGITS

digits	GR	our method
0-3	0.011±0.012	0.038±0.014
2-3	0.030±0.007	0.119±0.049
2-4	0.015±0.015	0.063±0.023
2-8	0.035±0.010	0.104±0.038
3-5	0.037±0.009	0.101±0.039
3-8	0.018±0.014	0.051±0.019
4-7	0.024±0.010	0.065±0.028
4-9	0.020±0.013	0.039±0.014
5-6	0.028±0.008	0.092±0.035
5-8	0.013±0.013	0.034±0.011
7-9	0.023±0.014	0.073±0.034
8-9	0.010±0.010	0.021±0.007

compute the distances between $\mathbf{x}$ and all centers $\mathbf{t}_i$'s in order to select the neighbors that are within a distance of $\tau = 3h$.

As can be seen from Table I, performing mean shift on the original Parzen window density estimator, even with the use of kd-trees, is still slower than the proposed approach. Table II shows a more detailed breakdown of the time consumed. As can be seen, the kd-tree-based mean shift algorithm spends more time on clustering (i.e., the mean shift iterations) than on range searching. Therefore, even though the kd-tree can speed up range searching, the bottleneck is still on the mean shift iterations where a large number of kernels within the neighborhood domain (around 3000 as shown in Table II) have to be summed up at each iteration. In other words, for this clustering application, directly simplifying the data representation is more useful than improving the data indexing efficiency.

C. SVM Testing

In this section, we perform experiments on simplifying the SVM decision function. As discussed in Section IV, we use a spherical covariance for the simplified mixture components. The first data set is the USPS database[5] for digit recognition. It contains 7291 training samples (16×16 gray image) and 2007 test samples. Here, we perform binary classification on some difficult digit pairs with relatively high classification errors (larger than 1%). For each classification task, we first use cross validation to choose the regularization parameter C and the γ parameter of the radial basis function (RBF) kernel $\exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\gamma)$. The obtained parameter values are listed in Table III. Then we simplify the obtained SVM decision function using GR and our method. Note that we first decompose the SVM decision function into two submodels, one with components having positive $\alpha_i y_i$'s and the other with components having negative $\alpha_i y_i$'s. For each submodel, we then use $m = \lfloor 2.5\% \text{ of } n \rfloor$ components for approximation, where n is the total number of support vectors. Therefore, as is also shown in Table III, the actual number of components (m) is usually smaller than 5% of n. Each classification task is repeated 100 times. For each run, both the GR and our methods use the same initial partitioning, which is obtained by the weighted k-means using random seeds on the support vectors and the multipliers α_i's as weights.

Results on the testing error are shown in Table IV. Statistical significance test (via the paired t-test) is also performed on the difference between the testing errors obtained by GR and our method. A positive t-value indicates that GR has a larger testing error. As can be seen, the proposed method is more accurate than GR. Table V compares the time consumptions of the two methods. As can be seen, the GR method is only faster than our approach by two to four times.

For more benchmark comparisons, we experiment with data sets from the LibSVM data archive[6] (Table VI). Some of these data sets (including `svmgd1a`, `splice`, `w1a`, and `adult1a`) have the training and testing splits available. We first use 20% of the training data as validation set for determining the C and

[5]ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/

[6]http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

TABLE VI
DATA SETS FROM THE LIBSVM DATA ARCHIVE AND THE PARAMETERS USED

data	training set	test set	dim	γ	C	(average) #SV (n)	(average) m
svmgd1a	3089	4000	4	5824.19	1	359	16
w1a	2477	1000	300	101.82	100	323	16
splice	1000	2175	60	143.59	10	584	28
adult1a	1605	1000	123	61.48	10	641	32
german	1000	-	24	13.18	1	482.4±9.5	23.2±0.9
pima	768	-	8	2.07	10	321.2±7.8	15.2±0.9
breastcr	683	-	10	7.02	1	61.3±4.1	2.0±0.0
australian	690	-	14	50.54	10	181.3±7.0	8.0±0.0
heart	270	-	13	29.67	1	110.8±4.1	4.0±0.2
liver	345	-	6	4.74	100	179.5±5.5	8.0±0.0
diabetes	768	-	8	4.13	1	350.4±7.1	16.2±0.6
ionosph	351	-	33	2.36	1	159.2±4.4	15.0±1.0
sonar	208	-	60	10.34	10	114.5±3.7	10.2±0.5

TABLE VII
TESTING ERRORS (IN PERCENT) ON THE LIBSVM DATA SETS

data	original	GR	ours	t (confidence)
svmgd1a	3.32	4.82±1.75	3.600±0.30	7.29(99.9%)
w1a	2.200	4.41±7.38	2.38±0.348	2.74(99.5%)
splice	10.34	12.37±0.66	11.21±0.59	19.64(99.9%)
adult1a	14.50	14.19±0.52	14.22±0.37	-0.59(—–)
german	24.50	26.81±3.14	25.83±2.81	4.26(99.9%)
pima	23.09±2.70	25.31±3.28	24.23±3.04	5.65(99.9%)
breastcnr	2.86±1.23	2.94±1.27	2.92±1.18	0.68(—–)
australian	14.98±2.79	14.96±2.81	15.05±2.90	-1.146(—–)
heart	17.09±4.53	16.64±4.50	16.68±4.63	-0.33(—–)
liver	27.50±5.04	21.85±13.96	21.07±11.92	1.22(—–)
diabetes	22.87±2.86	22.86±2.99	22.83±2.85	0.328(—–)
ionosph	5.45±2.44	13.63±4.54	12.85±4.82	1.46(90.0%)
sonar	11.50±4.32	24.92±7.94	20.47±6.98	7.01(99.9%)

TABLE VIII
TIME CONSUMPTION (IN SECONDS) ON THE LIBSVM DATA SETS

data	GR	ours
svmgd1a	0.012±0.013	0.051±0.017
w1a	0.045±0.017	0.063±0.022
splice	0.026±0.007	0.078±0.013
adult1a	0.045±0.010	0.095±0.028
german	0.017±0.012	0.052±0.023
pima	0.014±0.011	0.042±0.018
breastcnr	0.003±0.006	0.013±0.005
australian	0.005±0.007	0.016±0.006
heart	0.003±0.006	0.013±0.005
liver	0.004±0.008	0.017±0.010
diabetes	0.014±0.008	0.044±0.013
ionosph	0.007±0.008	0.032±0.012
sonar	0.007±0.008	0.025±0.008

γ parameters. GR and our method are then used to simplify the resultant decision function, and their prediction errors on the test data[7] are computed. We repeat both methods 100 times for each task due to the random initialization. For the other data sets that have only the training part provided, after we use the validation set to choose C and γ, we randomly split the data into training and testing parts with the ratio $4:1$ for 100 times, and for each repetition apply the two methods to simplify the SVM decision function. The data sets and their parameters are reported in Table VI. For most data sets, we choose the number of components m as 5% of the number of support vectors. However, for the `ionosph` and `sonar` data sets, we find that their decision functions are relatively difficult to approximate using a few components. So we choose m as 10% of the support vectors for these two data sets.

As can be seen from Table VII, by using only 5% of the support vectors, both methods achieve testing errors that are close to the original SVM classifier. The difference in performance between the two methods is not statistically significant on the `adult1a`, `breastcnr`, `australian`, `heart`, `liver`, and `diabetes` data sets; but for all the remaining data sets, our method is statistically significantly better than GR with a confidence level of at least 99.5%. Moreover, as can be seen from Table VIII, the GR method is only faster than our approach by two to four times.

To examine how the number of components affects the performance, we perform more experiments on the `ionosph` and `sonar` data sets whose decision functions are relatively difficult to approximate. We split them into fixed training and test sets, and train an SVM using the parameters reported in Table VI. In approximating the decision function, we gradually increase m from 5% to 35% of the number of support vectors, and report the testing errors of the two methods. For each m, we repeat the experiments 100 times using random initial start. The results are plotted in Fig. 3. As can be seen, both methods have improved performance when the number of components m increases.

VI. CONCLUSION

In this paper, we considered the problem of simplifying mixture models for prospective computational saving in a potentially large number of learning problems. We propose a new method to reduce model complexity by first grouping similar components into compact clusters and then performing local function approximation. Encouraging results are obtained in various tasks involving mixture models, such as kernel density estimation, clustering, and SVM testing.

Several problems remain to be explored in the future, for example, how to determine the optimal number of components in the simplified model, and how to use a combination of different types of kernels for function approximation. Our algorithm can be deemed as a preimage problem that allows variable kernel mapping. It would be interesting to study its performance in tasks like feature extraction and denoising. At last, we will consider how to utilize the parsimonious density model obtained through our method to scale up eigenvalue decomposition of kernel matrices, for which a preliminary work has been reported

[7]The test sets are quite large for `w1a` (47 272) and `adult1a` (30 956), so we randomly choose a testing subset of 1000 samples.

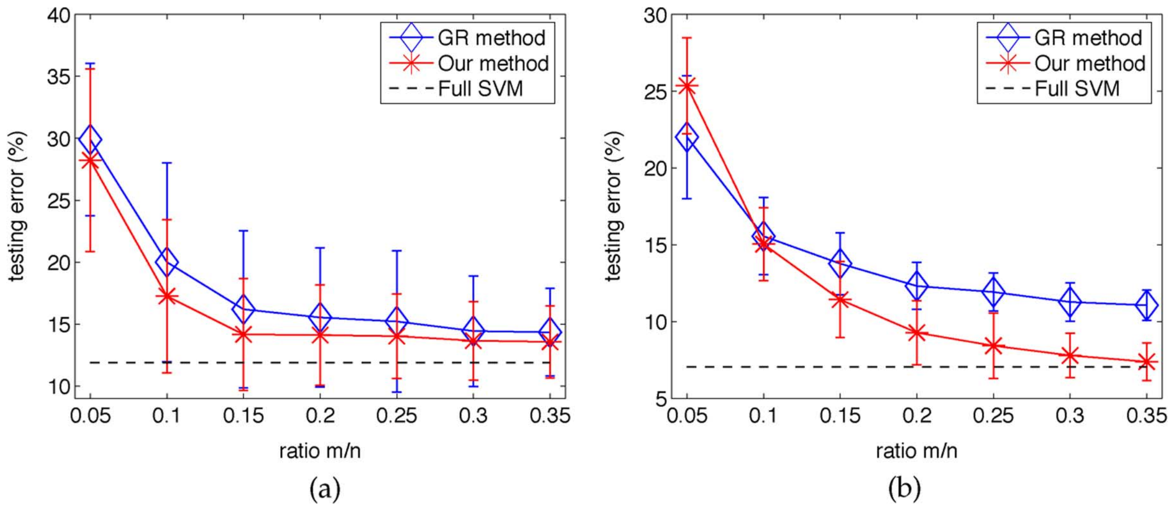

Fig. 3. Testing error versus the number of components used in our method and the GR method. (a) Sonar. (b) Ionosph.

in [42] that uses a simple divide-and-conquer strategy with normalized data histogram as the density model. This is potentially very useful for many learning algorithms such as spectral clustering, manifold learning, and dimension reduction.

APPENDIX I
PROOF OF PROPOSITION 1

First we introduce a new objective $\mathcal{Y} = \sum_{i=1}^{m} \mathcal{Y}_i$, where

$$\mathcal{Y}_i = \int \left(\mathcal{R}_i(\mathbf{x}) - \frac{1}{Z_i} \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x} \tag{32}$$

and $Z_i = \sum_{j \in S_i} \alpha_j$. Note that both $\mathcal{Q}$ (10) and $\mathcal{Y}$ (32) are decomposed into m independent terms under the L_2 distance. Therefore, it suffices to study the relationship for each pair of terms (denoted $\mathcal{Q}_i$ and $\mathcal{Y}_i$), both of which optimize over the representative $\mathcal{R}_i$. Note that $\mathcal{Q}_i$ can be written as

$$\begin{aligned}\mathcal{Q}_i(\mathcal{R}_i) &= \sum_{j \in S_i} \alpha_j \int (\phi_j(\mathbf{x}) - \mathcal{R}_i(\mathbf{x}))^2 \, d\mathbf{x} \\ &= \left(\sum_{j \in S_i} \alpha_j \int \phi_j^2(\mathbf{x}) d\mathbf{x} \right) + Z_i \int \mathcal{R}_i(\mathbf{x})^2 d\mathbf{x} \\ &\quad - 2 \sum_{j \in S_i} \int \alpha_j \phi_j(\mathbf{x}) \mathcal{R}_i(\mathbf{x}) d\mathbf{x}.\end{aligned} \tag{33}$$

The first term is independent of the optimization variable $\mathcal{R}_i$. On the other hand, $Z_i\mathcal{Y}_i$ can be written as

$$\begin{aligned}Z_i\mathcal{Y}_i(\mathcal{R}_i) &= Z_i \int \mathcal{R}_i(\mathbf{x})^2 d\mathbf{x} + \frac{1}{Z_i} \int \left(\sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x} \\ &\quad - 2 \sum_{j \in S_i} \int \alpha_j \phi_j(\mathbf{x}) \mathcal{R}_i(\mathbf{x}) d\mathbf{x}\end{aligned} \tag{34}$$

where the second term is also independent of the variable $\mathcal{R}_i$.

From (33) and (34), we can see that the two objectives $\mathcal{Q}_i$ and $Z_i\mathcal{Y}_i$ differ by only a constant that is independent of the $\mathcal{R}_i$ to be optimized. Therefore, the representative $\mathcal{R}_i^{\mathcal{Q}}(\mathbf{x})$ obtained by minimizing $\mathcal{Q}_i$ is the same as the representative $\mathcal{R}_i^{\mathcal{Y}}(\mathbf{x})$ obtained by minimizing $\mathcal{Y}_i$. Write $\mathcal{R}_i$ for the commoner minimizer, i.e., $\mathcal{R}_i = \mathcal{R}_i^{\mathcal{Q}} = \mathcal{R}_i^{\mathcal{Y}}$. On the other hand, comparing (6) and (32), we can see that their optimal solutions are connected by $\tilde{R}_i = Z_i\mathcal{R}_i^{\mathcal{Y}}$. Therefore, we have $\tilde{R}_i = Z_i\mathcal{R}_i(\mathbf{x})$.

APPENDIX II
DERIVATION OF THE LOCAL MODEL-SIMPLIFICATION ERROR $\bar{\mathcal{E}}_i$ IN (15)

Using the notation of profiles, the gradient of the kernel can be written as $\partial_{\mathbf{x}} K_{\mathbf{H}}(\mathbf{x}) = 2k'(r)\mathbf{H}^{-1}\mathbf{x}$, where $r = \mathbf{x}^\top\mathbf{H}^{-1}\mathbf{x}$. In the following, we first introduce two lemmas.

Lemma 1: For $\mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^d$, and (nonsingular) positive-definite matrix $\mathbf{H} \in \mathbb{R}^{d\times d}$, we have $\int |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} = C_{Kd}$, where C_{Kd} does not depend on $\mathbf{H}$ but only on the kernel K and dimensionality d.

Proof: By using the transform $\mathbf{y} = \mathbf{H}^{-1/2}(\mathbf{x} - \mathbf{x}_0)$, we have by the change of variables rule for integrals, $\int |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} = |\mathbf{H}|^{-1/2} \int K(\mathbf{H}^{-1/2}(\mathbf{x} - \mathbf{x}_0)) d\mathbf{x} = |\mathbf{H}|^{-1/2} \int K(\mathbf{y}) |\mathbf{H}|^{1/2} d\mathbf{y} = \int K(\mathbf{y}) d\mathbf{y}$, which is a constant depending only on the kernel K and dimensionality d. □

Lemma 2: For $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, and (nonsingular) positive-definite $\mathbf{H}_1, \mathbf{H}_2 \in \mathbb{R}^{d\times d}$, the following holds:

$$\begin{aligned}(\mathbf{x} - \mathbf{x}_1)^\top \mathbf{H}_1^{-1}(\mathbf{x} - \mathbf{x}_1) + (\mathbf{x} - \mathbf{x}_2)^\top \mathbf{H}_2^{-1}(\mathbf{x} - \mathbf{x}_2) \\ = (\mathbf{x} - \mathbf{z})^\top \mathbf{H}^{-1}(\mathbf{x} - \mathbf{z}) + r\end{aligned} \tag{35}$$

where

$$\mathbf{H} = \left(\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}\right)^{-1} \tag{36}$$

$$\mathbf{z} = \mathbf{H}\left(\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{H}_2^{-1}\mathbf{x}_2\right) \tag{37}$$

$$r = (\mathbf{x}_1 - \mathbf{x}_2)^\top (\mathbf{H}_1 + \mathbf{H}_2)^{-1}(\mathbf{x}_1 - \mathbf{x}_2). \tag{38}$$

Proof: The left-hand side of (35) can be written as

$$\begin{aligned}\mathbf{x}^\top(\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1})\mathbf{x} - 2(\mathbf{x}^\top\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}^\top\mathbf{H}_2^{-1}\mathbf{x}_2) \\ + \mathbf{x}_1^\top\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}_2^\top\mathbf{H}_2\mathbf{x}_2.\end{aligned} \tag{39}$$

The right-hand side of (35) can be written as

$$\begin{aligned}\mathbf{x}^\top\mathbf{H}^{-1}\mathbf{x} - 2\mathbf{x}^\top\mathbf{H}^{-1}\mathbf{z} + \mathbf{z}^\top\mathbf{H}^{-1}\mathbf{z} + r\\ = \mathbf{x}^\top(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})\mathbf{x} - 2(\mathbf{x}^\top\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}^\top\mathbf{H}_2^{-1}\mathbf{x}_2)\\ + \mathbf{z}^\top\mathbf{H}^{-1}\mathbf{z} + r \qquad (40)\end{aligned}$$

where we used the simple equality $\mathbf{H}^{-1} = \mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}$ by (36) and $\mathbf{H}^{-1}\mathbf{z} = \mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{H}_2^{-1}\mathbf{x}_2$ by (37). Comparing (39) and (40), we only need to prove

$$\mathbf{x}_1^\top\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}_2^\top\mathbf{H}_2\mathbf{x}_2 = \mathbf{z}^\top\mathbf{H}^{-1}\mathbf{z} + r. \qquad (41)$$

The right-hand side of (41) can be reduced as

$$\begin{aligned}&\mathbf{z}^\top\mathbf{H}^{-1}\mathbf{z} + r\\ &= \mathbf{x}_1^\top\mathbf{H}_1^{-1}(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_1^{-1}\mathbf{x}_1\\ &\quad + 2\mathbf{x}_1^\top\mathbf{H}_1^{-1}(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_2^{-1}\mathbf{x}_2\\ &\quad + \mathbf{x}_2^\top\mathbf{H}_2^{-1}(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_2^{-1}\mathbf{x}_2 + \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_1\\ &\quad - 2\mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_2 + \mathbf{x}_2^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_2.\end{aligned}$$

By using $(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1} = \mathbf{H}_1(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{H}_2 = \mathbf{H}_2(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{H}_1$ and plugging it into the obvious place in the above equation, $\mathbf{z}^\top\mathbf{H}^{-1}\mathbf{z} + r$ can be further reduced to

$$\begin{aligned}&\mathbf{x}_1^\top\mathbf{H}_1^{-1}(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_1^{-1}\mathbf{x}_1\\ &\quad + \mathbf{x}_2^\top\mathbf{H}_2^{-1}(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_2^{-1}\mathbf{x}_2\\ &\quad + \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_1 + \mathbf{x}_2^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_2\\ &\quad = \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{H}_2\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_1\\ &\qquad + \mathbf{x}_2^\top(\mathbf{H}_1^{-1}+\mathbf{H}_2^{-1})^{-1}\mathbf{H}_1\mathbf{H}_2^{-1}\mathbf{x}_2\\ &\qquad + \mathbf{x}_2^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}\mathbf{x}_2\\ &\quad = \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}(\mathbf{H}_2\mathbf{H}_1^{-1}+\mathbf{I})\mathbf{x}_1\\ &\qquad + \mathbf{x}_2^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}(\mathbf{H}_1\mathbf{H}_2^{-1}+\mathbf{I})\mathbf{x}_2\\ &\quad = \mathbf{x}_1^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}(\mathbf{H}_1+\mathbf{H}_2)\mathbf{H}_1^{-1}\mathbf{x}_1\\ &\qquad + \mathbf{x}_2^\top(\mathbf{H}_1+\mathbf{H}_2)^{-1}(\mathbf{H}_1+\mathbf{H}_2)\mathbf{H}_2^{-1}\mathbf{x}_2\\ &\quad = \mathbf{x}_1^\top\mathbf{H}_1^{-1}\mathbf{x}_1 + \mathbf{x}_2^\top\mathbf{H}_2^{-1}\mathbf{x}_2\end{aligned}$$

which is exactly the left-hand side of (41). Therefore, (41) holds and Lemma 2 is proved based on our previous arguments. □

Using Lemma 2 and (14), the multiplication of two kernels can be simplified as

$$\begin{aligned}&K_{\mathbf{H}_1}(\mathbf{x}-\mathbf{x}_1)K_{\mathbf{H}_2}(\mathbf{x}-\mathbf{x}_2)\\ &\quad = k((\mathbf{x}-\mathbf{x}_1)^\top\mathbf{H}_1^{-1}(\mathbf{x}-\mathbf{x}_1) + (\mathbf{x}-\mathbf{x}_2)^\top\mathbf{H}_2^{-1}(\mathbf{x}-\mathbf{x}_2))\\ &\quad = k(r)K_{\mathbf{H}}(\mathbf{x}-\mathbf{z}) \qquad (42)\end{aligned}$$

with $\mathbf{H}, \mathbf{z}$, and r given in (36), (37), and (38), respectively. Now we begin to derive $\bar{\mathcal{E}}_i$ (15). Using (6), we have

$$\bar{\mathcal{E}}_i = \int\left(\sum_{j\in S_i}\alpha_j\phi_j(\mathbf{x})\right)^2 d\mathbf{x} + \int w_i^2 g_i^2(\mathbf{x})d\mathbf{x} - 2\int w_i g_i(\mathbf{x})\sum_{j\in S_i}\alpha_j\phi_j(\mathbf{x})d\mathbf{x}.$$

The first term is independent of the unknowns. The second term can be rewritten as

$$\begin{aligned}&\int w_i^2 g_i^2(\mathbf{x})d\mathbf{x}\\ &\quad = \int w_i^2 C_{Kd}^{-2}|\tilde{\mathbf{H}}_i|^{-1}K_{\tilde{\mathbf{H}}_i/2}(\mathbf{x}-\mathbf{t}_i)\,d\mathbf{x}\\ &\quad = w_i^2 C_{Kd}^{-1}|2\tilde{\mathbf{H}}_i|^{-1/2}\int C_{Kd}^{-1}|\tilde{\mathbf{H}}_i/2|^{-1/2}K_{\tilde{\mathbf{H}}_i/2}(\mathbf{x}-\mathbf{t}_i)d\mathbf{x}\\ &\quad = w_i^2 C_{Kd}^{-1}|2\tilde{\mathbf{H}}_i|^{-1/2}.\end{aligned}$$

For the third term

$$\begin{aligned}&2\int w_i g_i(\mathbf{x})\sum_{j\in S_i}\alpha_j\phi_j(\mathbf{x})d\mathbf{x}\\ &\quad = \sum_{j\in S_i}\int\frac{2w_i\alpha_j C_{Kd}^{-2}}{|\tilde{\mathbf{H}}_i\mathbf{H}_j|^{1/2}}K_{\tilde{\mathbf{H}}_i}(\mathbf{x}-\mathbf{t}_i)K_{\mathbf{H}_j}(\mathbf{x}-\mathbf{x}_j)d\mathbf{x}\\ &\quad = \sum_{j\in S_i}\frac{2w_i\alpha_j k(r_{ij})C_{Kd}^{-1}}{|\tilde{\mathbf{H}}_i\mathbf{H}_j|^{1/2}}\int C_{Kd}^{-1}K_{\mathbf{H}_{ij}}(\mathbf{x}-\mathbf{z}_{ij})d\mathbf{x}\end{aligned}$$

where

$$\begin{aligned}\mathbf{H}_{ij} &= \left(\tilde{\mathbf{H}}_i^{-1}+\mathbf{H}_j^{-1}\right)^{-1},\\ r_{ij} &= (\mathbf{x}_j-\mathbf{t}_i)^\top(\tilde{\mathbf{H}}_i+\mathbf{H}_j)^{-1}(\mathbf{x}_j-\mathbf{t}_i),\\ \mathbf{z}_{ij} &= \left(\tilde{\mathbf{H}}_i^{-1}+\mathbf{H}_j^{-1}\right)^{-1}\left(\tilde{\mathbf{H}}_i^{-1}\mathbf{t}_i+\mathbf{H}_j^{-1}\mathbf{x}_j\right). \qquad (43)\end{aligned}$$

From Lemma 1, $\int C_{Kd}^{-1}K_{\mathbf{H}_{ij}}(\mathbf{x}-\mathbf{z}_{ij})d\mathbf{x} = |\mathbf{H}_{ij}|^{1/2}$, so we further reduce it to

$$\begin{aligned}&2\int w_i g_i(\mathbf{x})\sum_{j\in S_i}\alpha_j\phi_j(\mathbf{x})d\mathbf{x}\\ &\quad = \sum_{j\in S_i}\frac{2w_i\alpha_j k(r_{ij})C_{Kd}^{-1}}{|\tilde{\mathbf{H}}_i\mathbf{H}_j|^{\frac{1}{2}}}|\mathbf{H}_{ij}|^{\frac{1}{2}}\\ &\quad = 2w_i\sum_{j\in S_i}\frac{\alpha_j C_{Kd}^{-1}k(r_{ij})}{|\tilde{\mathbf{H}}_i+\mathbf{H}_j|^{\frac{1}{2}}}.\end{aligned}$$

By combining all three terms, we obtain (15).

Appendix III
Derivation of $D(\phi_j, \mathcal{R}_i)$

According to Proposition 1, $D(\phi_j,\mathcal{R}_i) = D(\phi_j,(w_i g_i)/(Z_i))$. Moreover, (15) is equal to $D(w_i g_i, \sum_{j\in S_i}\alpha_j\phi_i)$. Therefore, $D(\phi_j,\mathcal{R}_i)$ can be deemed as a simpler version of (15), and can be obtained easily by setting the following in (15): 1) $w_i \leftarrow w_i/Z_i$; 2) $j \leftarrow j \in S_i$; and 3) $\alpha_j \leftarrow 1$.

Appendix IV
Matrix Derivatives

In differentiating the objective function ($\bar{\mathcal{E}}_i$) w.r.t. the variables ($\mathbf{t}_i$ and $\tilde{\mathbf{H}}_i$), we need to use rules from matrix calculus.

For the readers' convenience, we list in the following several rules that may not be well known but play an important role in our derivations. For more rules on matrix derivatives, interested readers are referred to [27].

1) (Chain rule) For any $\mathbf{X} \in \mathbb{R}^{m \times n}$, and $y = f(\mathbf{X}), g(y)$ are differentiable real-valued functions, then

$$\frac{\partial g(f(\mathbf{X}))}{\partial \mathbf{X}} = \frac{dg(y)}{dy} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}.$$

2) Derivative of determinant: For any nonsingular $\mathbf{X} \in \mathbb{R}^{m \times m}$

$$\frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = |\mathbf{X}|(\mathbf{X}^{-1})'.$$

3) For any nonsingular $\mathbf{X} \in \mathbb{R}^{m \times m}$ and $\mathbf{a}, \mathbf{b} \in \mathbb{R}^m$

$$\frac{\partial \mathbf{a}'\mathbf{X}^{-1}\mathbf{b}}{\partial \mathbf{X}} = (\mathbf{X}^{-1})'\mathbf{a}\mathbf{b}'(\mathbf{X}^{-1})'.$$

REFERENCES

[1] G. A. Babich and O. I. Camps, "Weighted Parzen windows for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 567–570, May 1996.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *J. Mach. Learn. Res.*, vol. 6, pp. 1705–1749, 2005.

[3] C. J. C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector machines," in *Advances in Neural Information Processing Systems 9*. Cambridge, MA: MIT Press, 1996, pp. 375–381.

[4] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 281–288, Feb. 2003.

[5] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proc. Int. Conf. Comput. Vis.*, 1999, vol. 2, pp. 1197–1204.

[6] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[8] D. Comaniciu, V. Ramesh, and P. Meer, "The variable bandwidth mean shift and data-driven scale selection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, vol. 1, pp. 438–445.

[9] J. V. Davis and I. Dhillon, "Differential entropic clustering of multivariate Gaussians," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press, 2007.

[10] D. DeMenthon and D. Doermann, "Video retrieval using spatio-temporal descriptors," in *Proc. 11th ACM Int. Conf. Multimedia*, Berkeley, CA, 2003, pp. 508–517.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[12] J. Fan and J. S. Marron, "Fast implementations of nonparametric curve estimators," *J. Comput. Graph. Statist.*, vol. 3, no. 1, pp. 35–56, Mar. 1994.

[13] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975.

[14] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions, a texture classification example," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, pp. 456–463.

[15] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.

[16] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[17] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1253–1264, Oct. 2003.

[18] J. Goldberger, H. Greenspan, and J. Dreyfuss, "Simplifying mixture models using the unscented transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1496–1502, Aug. 2008.

[19] J. Goldberger and S. Roweis, "Hierarchical clustering of a mixture model," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 505–512.

[20] B. Han, D. Comaniciu, Y. Zhu, and L. Davis, "Incremental density approximation and kernel-based Bayesian filtering for object tracking," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2004, vol. 1, pp. 638–644.

[21] W. Härdle and D. W. Scott, "Smoothing by weighted averaging of rounded points," *Comput. Stat.*, vol. 7, pp. 97–128, 1992.

[22] A. J. Izenman, "Recent developments in nonparametric density estimation," *J. Amer. Stat. Assoc.*, vol. 86, no. 413, pp. 205–224, 1991.

[23] B. Jeon and D. A. Landgrebe, "Fast Parzen density estimation using clustering based branch and bound," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 950–954, Sep. 1994.

[24] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 477–482, Aug. 2002.

[25] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[26] K. I. Kim, K. Jung, and J. H. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1631–1639, Dec. 2003.

[27] H. Lütkepohl, *Handbook of Matrices*. New York: Wiley, 1996.

[28] J. S. Marron and M. P. Wand, "Exact mean integrated squared error," *Ann. Stat.*, vol. 20, no. 2, pp. 712–736, 1992.

[29] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 1992.

[30] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.

[31] A. W. Moore, "Very fast EM-based mixture model clustering using multiresolution kd-trees," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. San Mateo, CA: Morgan Kaufmann, 1998, pp. 543–549.

[32] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, vol. 33, pp. 1065–1075, 1962.

[33] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[34] K. Roeder, "Density estimation with confidence sets exemplified by superclusters and voids in the galaxies," *J. Amer. Stat. Assoc.*, vol. 85, no. 411, pp. 617–624, 1990.

[35] H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Boston, MA: Addison-Wesley, 1990.

[36] B. Schölkopf and A. J. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[37] D. W. Scott and S. J. Sheather, "Kernel density estimation with binned data," *Commun. Stat. A—Theory Methods*, vol. 14, pp. 1353–1359, 1985.

[38] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Distributed occlusion reasoning for tracking with nonparametric belief propagation," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005, pp. 1369–1376.

[39] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 605–612.

[40] P. Vincent and Y. Bengio, "Manifold Parzen windows," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 825–832.

[41] J. Zhang, K. Zhang, X. Xu, C. K. Tse, and M. Small, "Seeding the kernels in graphs: Toward multi-resolution community analysis," *New J. Phys.*, vol. 11, 2009, 113003.

[42] K. Zhang and J. T. Kwok, "Density-weighted Nyström method for computing large kernel eigensystems," *Neural Comput.*, vol. 21, no. 1, pp. 121–146, 2009.

[43] K. Zhang and J. T. Kwok, "Simplifying mixture models through function approximation," in *Advances in Neural Information Processing Systems 17*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. Cambridge, MA: MIT Press, 2007, pp. 1577–1584.

[44] K. Zhang, M. Tang, and J. T. Kwok, "Applying neighborhood consistency for fast clustering and kernel density estimation," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 2, pp. 1001–1007.

[45] X. S. Zhou, D. Comaniciu, and A. Gupta, "An information fusion framework for robust shape tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 1, pp. 115–129, Jan. 2005.

Kai Zhang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2008.

Then, he joined Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA. His research interest is machine learning and pattern recognition, in particular, large scale unsupervised learning and dimension reduction algorithms. He also works on applications in data mining, bioinformatics, and complex networks.

James T. Kwok (M'98–SM'07) received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 1996.

He then joined the Department of Computer Science, Hong Kong Baptist University, as an Assistant Professor. He returned to the Hong Kong University of Science and Technology in 2000, where he is now an Associate Professor at the Department of Computer Science and Engineering. His research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok is an Associate Editors of the IEEE TRANSACTIONS ON NEURAL NETWORKS and *Neurocomputing*. He received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006.