

Simplifying Mixture Models through Function Approximation

Kai Zhang
James T. Kwok

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

TWINSEN@CSE.UST.HK
JAMESK@CSE.UST.HK

Editor:

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 first grouping similar components together and then performing local fitting through function approximation. By using the squared loss to measure the distance between mixture models, our algorithm naturally combines the two different tasks of *component clustering* and *model simplification*. The proposed method can be used to speed up various algorithms that use mixture models during training (e.g., Bayesian filtering, belief propagation) or testing (e.g., kernel density estimation, SVM testing). Encouraging results are observed in the experiments on density estimation, clustering-based image segmentation and simplification of SVM decision functions.

Keywords: Mixture models, Clustering, KL divergence, Squared loss

Correspondence author: Kai Zhang

1. Introduction

In data analysis, it is often useful to obtain a probability density estimate for a set of 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 (Izenman, 1991). In this paper, we focus on the *finite mixture model* (McLachlan and Peel, 2000), which describes the distribution by a mixture of simple parametric functions $\phi(\cdot)$'s, as:

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

Here, $\phi_j(\mathbf{x})$ is the j th 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*. Other popular choices include the Bernoulli, Poisson, or circular normal distributions. Finite mixture models have proven to be very powerful in modeling complex, non-Gaussian distributions (Roeder, 1990; Marron and Wand, 1992). It is widely used in classification, clustering and density estimation (McLachlan, 1992), and in applications such as speech processing (Rabiner, 1989). Typically, the model parameters are estimated by the Expectation Maximization (EM) algorithm (Dempster et al., 1977).

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. For example, consider the Parzen window density estimator (Parzen, 1962)

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

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 under 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) (Schölkopf and Smola, 2002)

$$f(\mathbf{x}) = \sum_{j=1}^{S_v} \alpha_j K(\mathbf{x}, \mathbf{x}_j) + b.$$

In case of the Gaussian kernel K , the decision function is very similar in form to the Gaussian mixture model. How to reduce a non-sparse 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 (Burges and Schölkopf, 1996). Other examples that involve mixture models include the particle filter (Han et al., 2004) and non-parametric belief propagation (Sudderth et al., 2003, 2005), 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.

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. Therefore, unnecessary component summations in the model evaluation can be circumvented by performing a neighborhood search through spatial data structures, such as the kd-trees (Kanungo et al., 2002; Moore, 1998). However, such algorithms may scale poorly with the input dimensionality. Another approach, first introduced by Scott and Sheather (1985) in the context of kernel density estimation, pre-bins the data into an 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 (Scott and Sheather, 1985; Härdle and Scott, 1992; Fan and Marron, 1994) are focused on univariate data. Babich and Camps (1996) and Jeon and Landgrebe (1994) perform pre-clustering of the data and then use the cluster centers as a reduced sample set. However, they are based on heuristic procedures which lack precise approximation criteria. Girolami and He (2003) proposed another reduced set density estimator that uses quadratic programming (QP) to obtain a sparse representation. However, the resultant QP problem still scales cubically with the sample size. Even with the use of efficient optimization techniques like sequential minimal optimization (SMO) (Platt, 1999), the empirical complexity is still $O(n^2)$ (Girolami and He, 2003). Recently, Goldberger and Roweis (2005) and Davis and Dhillon (2007) 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 (Goldberger and Roweis, 2005), sensor networks and statistical debugging (Davis and Dhillon, 2007).

In this paper, we propose to simplify the mixture model by first grouping similar components and then performing local fitting through function approximation. With the use of the L_2 distance measure, we show that both steps are indeed minimizing upper bounds of the exact model simplification error. In other words, component grouping and model simplification are naturally combined together. We organize the paper as follows. Section 2 describes the proposed algorithm. We first provide a big picture of the algorithm and analyze the relationship between the two underlying steps. Then we present the detailed mathematical derivations (Section 2.1). Section 3 gives more in-depth discussions, including comparisons with related approaches (Section 3.1), some interesting properties of our approach (Sections 3.2 and 3.3), and the computational complexity analysis (Section 3.4). In Section 4, we discuss the application of our approach in simplifying the SVM decision function. The performance of the proposed method is evaluated through a number of experiments in Section 5, and the last section gives some concluding remarks. Preliminary results have been reported in the conference version (Zhang and Kwok, 2007).

2. Approximation Algorithm

Given a mixture model f (1), we assume that the j th component takes the form

$$\phi_j(\mathbf{x}) = C_{Kd}^{-1} |\mathbf{H}_j|^{-1/2} K_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j), \quad (3)$$

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 following conditions:

$$\begin{aligned} \int_{\mathbb{R}^d} K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} &= 1, & \lim_{\|\mathbf{x}\| \rightarrow \infty} \|\mathbf{x}\|^d K_{\mathbf{H}}(\mathbf{x}) &= 0, \\ \int_{\mathbb{R}^d} \mathbf{x} K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} &= 0, & \int \mathbf{x} \mathbf{x}^\top K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x} &= C_{Kd} \mathbf{I}. \end{aligned}$$

Here, $C_{Kd} = \int K_{\mathbf{H}}(\mathbf{x}) d\mathbf{x}$ is a normalization factor that depends on the kernel $K_{\mathbf{H}}$ and 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 argument. For the Gaussian kernel, $k(r) = \exp(-\frac{1}{2}r)$; for the Epanechnikov kernel, $k(r) = \begin{cases} \frac{3}{4}(1-r) & r < 1 \\ 0 & r \geq 1 \end{cases}$; whereas for the negative exponential kernel, $k(r) = \exp(-\beta r^{1/2})$.

Our goal is to approximate f with a simplified mixture model

$$g(\mathbf{x}) = \sum_{i=1}^m w_i g_i(\mathbf{x}), \quad (4)$$

where $m \ll n$. Each component g_i has a form similar to (3):

$$g_i(\mathbf{x}) = C_{Kd}^{-1} |\tilde{\mathbf{H}}_i|^{-1/2} K_{\tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{t}_i), \quad (5)$$

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 up to 1 and this can be achieved by re-scaling w_i 's with $1/\sum_{i=1}^m w_i$. Given a distance measure $D(\cdot, \cdot)$ defined between functions, the error induced by approximating f with g is

$$\mathcal{E} = D(f, g) = D\left(\sum_{i=1}^m w_i g_i, \sum_{j=1}^n \alpha_j \phi_j\right). \quad (6)$$

Direct minimization of \mathcal{E} is usually difficult since both f and g are composed of multiple components. However, the problem can be much simplified by minimizing an upper bound of \mathcal{E} as follows. Consider the L_2 distance

$$D(\phi, \phi') = \int (\phi(\mathbf{x}) - \phi'(\mathbf{x}))^2 d\mathbf{x}.$$

Suppose that the mixture components $\{\phi_j\}_{j=1}^n$ are divided into disjoint clusters $\{S_1, \dots, S_m\}$. By using the Cauchy-Schwarz inequality

$$\left(\sum_{i=1}^m a_i\right)^2 \leq m \sum_{i=1}^m a_i^2, \quad (7)$$

The model simplification error \mathcal{E} (6) can be upper bounded by

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

Define

$$\bar{\mathcal{E}} = m \sum_{i=1}^m \bar{\mathcal{E}}_i = m \sum_{i=1}^m \int \left(w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x}. \quad (8)$$

We can see that the upper-bound $\bar{\mathcal{E}}$ is comprised of local model-simplification errors $\bar{\mathcal{E}}_i$'s. Given a fixed partition of the mixture components, $\bar{\mathcal{E}}$ can be minimized equivalently by minimizing the $\bar{\mathcal{E}}_i$'s (independent of each other). Therefore, minimizing $\bar{\mathcal{E}}$ will be much easier. A natural two-step procedure to minimize $\bar{\mathcal{E}}$ is to first group the mixture components into compact clusters (i.e., *component grouping*), and then find a good representative $w_i g_i$ (8) for each group (i.e., *local approximation*).

Component grouping. We perform component clustering in an iterative manner. Given an initial partitioning of the mixture components, $\{S_1, S_2, \dots, S_m\}$, we compute the representative component $\mathcal{R}_i(\cdot)$ for each S_i that minimizes the local quantization error

$$\mathcal{R}_i = \arg \min_{\mathcal{R}_i(\mathbf{x})} \sum_{j \in S_i} \alpha_j \int (\mathcal{R}_i(\mathbf{x}) - \phi_j(\mathbf{x}))^2 d\mathbf{x}. \quad (9)$$

Here, the representative \mathcal{R}_i takes the same parametric form of $w_i g_i$ in (5). Then, re-assign each ϕ_j to its closest representative w.r.t. the distance

$$D(\phi_j, \mathcal{R}_i) = \int (\mathcal{R}_i(\mathbf{x}) - \phi_j(\mathbf{x}))^2 d\mathbf{x}, \quad (10)$$

and repeat (9) and (10) until convergence. This is quite close in flavor to the vector quantization algorithm (Gersho and Gray, 1992). By doing this, we actually find a local minimum of the distortion error

$$Q = \sum_{i=1}^m \sum_{j \in S_i} \alpha_j D(\phi_j, \mathcal{R}_i). \quad (11)$$

Local approximation. After grouping the mixture components, we minimize the local model-simplification error (8) by

$$w_i g_i = \arg \min_{w_i g_i(\mathbf{x})} \int \left(w_i g_i(\mathbf{x}) - \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x}, \quad (12)$$

which can be performed independently for different clusters.

Recall that our ultimate goal is *model simplification*, i.e., approximating the original model f with a simpler model g , which is best measured by the model-simplification error \mathcal{E} (6). Therefore, it is natural to ask how these two steps are related to the final goal of model simplification and whether they are related themselves. In the following, we will prove that both steps minimize (different) upper bounds of the model-simplification error \mathcal{E} (6). More interestingly, their solutions, (9) and (12), are also closely related.

Proposition 1 *Let $D(\cdot, \cdot)$ be the L_2 distance between two functions. Suppose that the components ϕ_j 's have been partitioned into m clusters $\{S_1, S_2, \dots, S_m\}$, with the maximum cluster size being \bar{n} . Then component clustering and local approximation find local minima of*

$$\bar{\mathcal{E}} = m \sum_{i=1}^m D \left(w_i g_i(\mathbf{x}), \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right),$$

and

$$\tilde{\mathcal{E}} = m \bar{n} \sum_{i=1}^m \sum_{j \in S_i} \alpha_j^2 D(R_i^{\mathcal{Q}}(\mathbf{x}), \phi_j(\mathbf{x})),$$

respectively. Here, $R_i^{\mathcal{Q}}(\mathbf{x}) = \arg \min_R \sum_{j \in S_i} \alpha_j D(R(\mathbf{x}), \phi_j(\mathbf{x}))$, and $R(\mathbf{x})$ has the same functional form as (3). Moreover, both $\tilde{\mathcal{E}}$ and $\bar{\mathcal{E}}$ are upper bounds of the model-simplification error $\mathcal{E} = D \left(\sum_{i=1}^m w_i g_i(\mathbf{x}), \sum_{j=1}^n \alpha_j \phi_j(\mathbf{x}) \right)$, with

$$\mathcal{E} \leq \bar{\mathcal{E}} \leq \tilde{\mathcal{E}}.$$

Besides, the representatives obtained in the component clustering (\mathcal{R}_i in (9)) and local approximation steps ($w_i g_i$ in (12)) are related by

$$\mathcal{R}_i(\mathbf{x}) = w_i g_i(\mathbf{x}) / Z_i, \quad \text{where } Z_i = \sum_{j \in S_i} \alpha_j. \quad (13)$$

Proof is in Appendix A. 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. In other words, each iteration in component clustering (9) is actually performing a “re-normalized” version of the local approximation step (12). Therefore, from an algorithmic point of view, “component clustering” and “model simplification” are naturally combined together. After establishing the connection between \mathcal{R}_i (9) and $w_i g_i$ (12), we can derive the solution for one of them and then easily obtain the other. In the next subsection, we provide details on how to compute $w_i g_i$ (12).

2.1 Detailed Derivations

In this section, we show how to solve the optimization problem (12) in detail and obtain the local representative $w_i g_i$. Note that the specific form of the kernel (or, equivalently,

its profile) has to be determined before a concrete algorithm can be designed, or else the integration involved in the objectives (\mathcal{Q} and $\bar{\mathcal{E}}$) cannot be obtained in explicit forms. In the following, we will consider the Gaussian kernel. For other types of kernel, the derivations will differ but Proposition 1 still holds in general.

Using the notation of profiles, the gradient of the kernel can be represented as $\partial_{\mathbf{x}}K_{\mathbf{H}}(\mathbf{x}) = 2k'(r)\mathbf{H}^{-1}\mathbf{x}$, where $r = \mathbf{x}^\top\mathbf{H}^{-1}\mathbf{x}$. It is also easy to verify that

$$\begin{aligned} K_{\mathbf{H}_1}(\mathbf{x}_1) \cdot K_{\mathbf{H}_2}(\mathbf{x}_2) &= \exp\left(-\frac{1}{2}\mathbf{x}_1^\top\mathbf{H}_1\mathbf{x}_1\right) \exp\left(-\frac{1}{2}\mathbf{x}_2^\top\mathbf{H}_2\mathbf{x}_2\right) \\ &= \exp\left(-\frac{1}{2}\left(\mathbf{x}_1^\top\mathbf{H}_1\mathbf{x}_1 + \mathbf{x}_2^\top\mathbf{H}_2\mathbf{x}_2\right)\right), \end{aligned}$$

i.e.,

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

Then we can derive $\bar{\mathcal{E}}_i$ (8) as (detailed derivation is in Appendix B)

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

where

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

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

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

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 that is difficult to solve. Here, we decouple the relations among these three parameters and perform coordinate descent. Observe that $\bar{\mathcal{E}}_i$ is a quadratic function of w_i . Therefore, by choosing

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

and substituting it back into (15), we obtain the minimum of $\bar{\mathcal{E}}_i$ (for fixed values of \mathbf{t}_i and $\tilde{\mathbf{H}}_i$) as

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

The remaining task is to minimize $\bar{\mathcal{E}}_i^{\min}$ w.r.t. \mathbf{t}_i and $\tilde{\mathbf{H}}_i$. On setting $\partial_{\mathbf{t}_i} \bar{\mathcal{E}}_i^{\min} = 0$, we have

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

where

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

If $\tilde{\mathbf{H}}_i$ is fixed, we can obtain \mathbf{t}_i by starting with an initial $\mathbf{t}_i^{(0)}$, and then iterate (19) until convergence (recall that r_{ij} is a function of \mathbf{t}_i).

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

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

By using $\frac{1}{2} \tilde{\mathbf{H}}_i^{-1} - (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} = \frac{1}{2} (\mathbf{H}_j + \tilde{\mathbf{H}}_i)^{-1} (\mathbf{H}_j - \tilde{\mathbf{H}}_i) \tilde{\mathbf{H}}_i^{-1}$, we have

$$\tilde{\mathbf{H}}_i = \mathbf{P}_i^{-1} \sum_{j \in \mathcal{S}_i} \frac{\alpha_j (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i|^{1/2}} \left(k(r_{ij}) \mathbf{H}_j - 4k'(r_{ij}) (\mathbf{x}_j - \mathbf{t}_i) (\mathbf{x}_j - \mathbf{t}_i)^\top (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1} \tilde{\mathbf{H}}_i \right), \quad (20)$$

where

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

Together, \mathbf{t}_i and $\tilde{\mathbf{H}}_i$ are computed in an iterative manner. Given some initial choices of \mathbf{t}_i and $\tilde{\mathbf{H}}_i$, we alternate between fixing one variable and optimizing $\bar{\mathcal{E}}_i^{min}$ w.r.t. the other. We 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 the values of \mathbf{t}_i and $\tilde{\mathbf{H}}_i$, they are substituted into (17) to compute w_i . For initialization, we follow (Goldberger and Roweis, 2005) and set

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

which are the weighted mean and weighted covariance of the samples, respectively. Note that both can be computed efficiently. This initialization can be regarded as requiring (the normalized versions of) the local component group $\sum_{j \in \mathcal{S}_i} \alpha_j \phi_j$ and the reduced model $w_i g_i(\mathbf{x})$ (5) to have the same mean and covariance.

With the solution of $w_i g_i$ (12) in the local approximation step, the representative \mathcal{R}_i (9) in the component clustering step is also easily obtained by the connection stated in Proposition 1. The complete algorithm is presented in Algorithm 1. The detailed derivations of the distance function $D(\phi_j, \mathcal{R}_i)$ (line 22 in Algorithm 1) can be found in Appendix B.

Note that our objective function $\bar{\mathcal{E}}_i^{min}(\mathbf{t}_i, \tilde{\mathbf{H}}_i)$ is a (real-valued) function of the two variables, \mathbf{t}_i and $\tilde{\mathbf{H}}_i$. In our algorithm, we alternate between minimizing the objective

Input: $\{\alpha_j, \phi_j(\mathbf{x}_j, \mathbf{H}_j)\}_{j=1}^n$; initial partitioning of ϕ_j 's into $\{S_1, S_2, \dots, S_m\}$;
 $t = 0, e^{(t)} = 1e10; \epsilon_1 = 0.01, \epsilon_2 = 0.01$.

Output: $\{w_i, g_i(\mathbf{t}_i, \tilde{\mathbf{H}}_i)\}_{i=1}^m$.

```

1 repeat
2    $t = t + 1;$ 
3   for  $i = 1, 2, \dots, m$ . 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      $\tilde{\mathbf{H}}_i^{(0)} = \frac{1}{\sum_{j \in S_i} \alpha_j} \sum_{j \in S_i} \alpha_j (\mathbf{H}_j + (\mathbf{t}_i^{(0)} - \mathbf{x}_j)(\mathbf{t}_i^{(0)} - \mathbf{x}_j)^\top);$ 
6     repeat
7        $p = 0;$ 
8       repeat
9          $\mathbf{t}_i^{(p+1)} = \left( \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}^{p,0})(\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)})^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)}|^{1/2}} \right)^{-1} \sum_{j \in S_i} \frac{\alpha_j k'(r_{ij}^{p,0})(\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)})^{-1} \mathbf{x}_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(0)}|^{1/2}};$ 
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           $\tilde{\mathbf{H}}_i^{(q+1)} =$ 
15           $\left( \sum_{j \in S_i} \frac{(\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1} \alpha_j}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^2} k(r_{ij}^{p,q}) \right)^{-1} \sum_{j \in S_i} \frac{\alpha_j (\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1}}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^2} (k(r_{ij}^{p,q}) \mathbf{H}_j - 4k'(r_{ij}^{p,q})(\mathbf{x}_j - \mathbf{t}_i^{(p)})(\mathbf{x}_j - \mathbf{t}_i^{(p)})^\top) (\tilde{\mathbf{H}}_i^{(q)} + \mathbf{H}_j)^{-1} \tilde{\mathbf{H}}_i^{(q)};$ 
16           $q = q + 1;$ 
17          until  $\|\mathbf{H}_i^{(q)} - \mathbf{H}_i^{(q-1)}\|_F \leq \epsilon_2$ ;
18           $w_i = |2\tilde{\mathbf{H}}_i^{(q)}|^{1/2} \sum_{j \in S_i} \frac{\alpha_j k(r_{ij}^{p,q})}{|\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)}|^{1/2}};$ 
19        until  $p \leq 1$  and  $q \leq 1$ ;
20    end
21     $e^{(t)} = 0;$ 
22    for  $j = 1, 2, \dots, n$  do
23      for  $i = 1, 2, \dots, m$  do
24         $D(\phi_j, \mathcal{R}_i) = C_{Kd}^{-1} \left( |2\mathbf{H}_j|^{-1/2} + \frac{w_i^2}{Z_i^2} |2\tilde{\mathbf{H}}_i|^{-1/2} - 2\frac{w_i}{Z_i} k(r_{ij}^{p,q}) |\tilde{\mathbf{H}}_i + \mathbf{H}_j|^{-1/2} \right);$ 
25      end
26      Assign  $\phi_j$  to cluster  $S_i$  such that  $i = \arg \min_k D(\phi_j, \mathcal{R}_k)$ ;
27       $e^{(t)} = e^{(t)} + D(\phi_j, \mathcal{R}_i)$ ;
28    end
29  until  $|e^{(t)} - e^{(t-1)}| \leq 0.001 |e^{(t)}|;$ 

```

Algorithm 1: Algorithm for simplifying mixture models. Notations: $r_{ij}^{p,q} = (\mathbf{t}_i^{(p)} - \mathbf{x}_j)^\top (\mathbf{H}_j + \tilde{\mathbf{H}}_i^{(q)})^{-1} (\mathbf{t}_i^{(p)} - \mathbf{x}_j)$, $i, j, p, q \in \mathbb{Z}$, $\mathbf{t}^{(p)}, \mathbf{x}_j \in \mathbb{R}^{d \times 1}$, $\mathbf{H}_j, \tilde{\mathbf{H}}_i^{(q)} \in \mathbb{R}^{d \times d}$, Gaussian kernel $k(r) = \exp(-\frac{1}{2}r)$ and $k'(r) = -\frac{1}{2} \exp(-\frac{1}{2}r)$.

w.r.t. \mathbf{t}_i (steps 7-11) and $\tilde{\mathbf{H}}_i$ (steps 12-17), respectively, both of which have been shown to be gradient descent and convergent in our experimental setting. Therefore, our algorithm belongs to the general optimization framework of alternating optimization (Bezdek and Hathaway, 2003), which is defined as an iterative procedure for minimizing the objective jointly over all variables by alternating restricted minimizations over the individual subsets of variables. However, since the iteration formula in (19) and (20) may only converge to a local minimum, the convergence results in (Bezdek and Hathaway, 2003) cannot be applied here. Nevertheless, in practice we stop the algorithm by monitoring the objective value at each cycle of the outer loop (line 1-28). If the objective value decreases by more than a threshold when compared to the previous iteration, the algorithm continues; Otherwise, it terminates. Empirically, we never encountered an increase of the objective value in all our experiments.

3. Discussions

3.1 Comparison with Related Methods

There have been several related works on grouping the components in a mixture model. For example, Banerjee et al. (2005) 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 (11). Most related to ours is the work of (Goldberger and Roweis, 2005) 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 (4) by minimizing the following “local KL-distance”

$$d(f, g) = \sum_{j=1}^n \alpha_j KL(\phi_j || g_{\pi(j)}), \quad (21)$$

where $KL(\cdot || \cdot)$ is the KL-divergence, and $\pi(j)$ maps component ϕ_j to its closest component $g_{\pi(j)}$ (i.e., $\pi(j) = \arg \min_{i=1,2,\dots,m} KL(\phi_j || g_i)$). To minimize (21), an iterative procedure is applied which alternates between finding the partition and re-assigning the components to the closest representatives. At each iteration, the parameters of g_i (i.e., \mathbf{t}_i and $\tilde{\mathbf{H}}_i$) are determined as

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

Davis and Dhillon (2007) 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, both (Goldberger and Roweis, 2005) and (Davis and Dhillon, 2007) consider the problem of *component clustering*, i.e., grouping a set of Gaussian components ϕ_i 's, based on minimizing the distortion measure in (21). In comparison, our algorithm

attempts to solve the problem of *model simplification*, i.e., minimizing the distance between two mixture models f and g by optimizing its upper bound. Another difference is that (Goldberger and Roweis, 2005) and (Davis and Dhillon, 2007) adopt the KL-divergence in measuring the distance between Gaussian components, while we use the L_2 distance to measure the distance (upper bound) between two mixtures.

3.2 Choice of the Local Covariance

In this section, we discuss some interesting properties of our approximation scheme in Section 2.1. To have better intuitions, we examine the special case where the given mixture model f is a Parzen window density estimator (2), i.e., all ϕ_j 's have the same weight and covariance matrix ($\mathbf{H}_j = \mathbf{H}$ for $j = 1, 2, \dots, n$). Then, the bandwidth (20) of the representative kernel g_i can be decomposed as:

$$\tilde{\mathbf{H}}_i = \mathbf{H} + 4\tilde{\mathbf{H}}_i(\tilde{\mathbf{H}}_i + \mathbf{H})^{-1}\mathbf{V}_i, \quad (22)$$

where \mathbf{H} is the bandwidth of the original components, and

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

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 the Gaussian and the 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 scheme is similar to that used in the manifold Parzen windows (Vincent and Bengio, 2003), which is designed to handle sparse, high-dimensional data in a more robust manner. In comparison, related works on simplifying mixture models (Goldberger and Roweis, 2005; Davis and Dhillon, 2007) choose $\tilde{\mathbf{H}}_i = \mathbf{H} + Cov[S_i]$, where $Cov[S_i]$ is simply the (non-robust) covariance of S_i .

3.3 Relationship with Mean Shift

Mean shift (Comaniciu and Meer, 2002; Fukunaga and Hostetler, 1975) is an iterative mode-seeking algorithm widely used in pattern recognition and computer vision. Its basic assumption is that local maxima of the underlying density distribution correspond to clusters of the data. By shifting every data point iteratively along the (estimated) gradient of the density surface, dominant clusters of the data can be easily identified.

Given a sample set $\{x_i\}_{i=1}^n$, the Parzen window density estimator (2) is

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

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

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

This can be written as

$$2\mathbf{H}^{-1}\mathbf{x} \sum_{i=1}^n k'((\mathbf{x} - \mathbf{x}_i)' \mathbf{H}^{-1}(\mathbf{x} - \mathbf{x}_i)) = 2\mathbf{H}^{-1} \sum_{i=1}^n k'((\mathbf{x} - \mathbf{x}_i)' \mathbf{H}^{-1}(\mathbf{x} - \mathbf{x}_i)) \mathbf{x}_i,$$

or in iterative form (Comaniciu and Meer, 2002)

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

It has been shown (Comaniciu and Meer, 2002) that the 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 any prior assumption on the shape of the clusters, and the number of classes can be determined automatically based on the converged sample set. It has demonstrated great success in many vision applications, such as image segmentation (Comaniciu and Meer, 1999, 2002), tracking (Comaniciu et al., 2003; Han et al., 2004; Zhou et al., 2005) and video processing (DeMenthon and Doermann, 2003; Kim et al., 2003).

Considering the special case of fitting Parzen window density estimator as in Section 3.2, the iteration (19) for determining center \mathbf{t}_i of the representative g_i is reduced to¹

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

On 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 if the covariance matrices \mathbf{H}_j 's associated with each component is different, then mean shift procedure needs to be derived by computing the derivative of the following variable-bandwidth density estimator (Comaniciu, 2003; D. Comaniciu et al., 2001)

$$\hat{f} = \frac{1}{n(2\pi)^{d/2}} \sum_{j=1}^n |\mathbf{H}_j|^{-1/2} K_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j). \quad (27)$$

Here, $\mathbf{x}_j \in \mathbb{R}^d$, and $\mathbf{H}_j \in \mathbb{R}^{d \times d}$ is the covariance matrix associated with the j th component. This is actually the same as the iterations (19) for finding the component centers \mathbf{t}_i 's². Actually, in deriving our iteration (19) we follow exactly the same details (Section 2 of Comaniciu (2003)) to obtain the update equations in the form of fixed point iterations.

-
1. In the more general case where the covariances \mathbf{H}_j 's of the original mixture components are all different, the iteration (19) used to find the center of the representative local component g_i is known as the *variable bandwidth mean shift* (Comaniciu, 2003).
 2. Although the component here is uniformly weighted, it can still incorporate the non-uniform weighting situation by placing the number of samples at each \mathbf{x}_i that is equal to the weight.

3.4 Time Complexity

In this section, we consider the time complexity of the proposed algorithm. As can be seen from Algorithm 1, the largest loop is indexed by the time step t (line 1 to line 28, T steps). In this loop, we will treat altogether m clusters S_1, S_2, \dots, S_m . For each cluster, we first initialize \mathbf{t}_i and $\tilde{\mathbf{H}}_i$ (line 4~5), and then alternate between the update of \mathbf{t}_i (line 8~11, p steps) and the update of $\tilde{\mathbf{H}}_i$ (line 13~16, q steps), inside the loop line 6~19 (for S times). When both p and q are below a small integer (1 in our algorithm), it means that the alteration between updating \mathbf{t}_i and $\tilde{\mathbf{H}}_i$ converges and we can stop this loop. Then, we compute the distance between each component ϕ_j and the representatives \mathcal{R}_i 's (line 20~27), update the clustering by assigning ϕ_j to the closest representative \mathcal{R}_i , and start the next iteration.

Let n denotes the number of components in the original mixture, m is the number of clusters, n_i is the size of cluster i , and d is the dimensionality. So the overall complexity is

$$T \left(S \sum_{i=1}^m n_i d^3 (p + q) + n m d^3 \right) = T d^3 n (S(p + q) + m).$$

For computational efficiency, we can use diagonal or even spherical³ bandwidth matrices. If the covariances of the mixture components are also diagonal (as is commonly seen in kernel density estimators, SVM testing or belief propagation (Sudderth et al., 2003)), then the d^3 term in the complexity will be reduced to linear term d , and q will be equal to 1 because there exists a closed form, one-shoot solution for the iteration between line 13 and line 16 (as shown in (30)). In practice, it is observed that the number of iterations S, T, p, q are typically small numbers. In particular, we find that the largest loops T is around 15; S does not quite influence the performance. Therefore in practice, we usually set S as 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 largest loops for computational efficiency.

The algorithm in (Goldberger and Roweis, 2005; Davis and Dhillon, 2007) has a complexity of $O(lmnd^3)$ when g_i 's have full covariances, and $O(lmnd)$ when both g_i 's and mixture components have diagonal covariances. Empirically, it is faster than our approach since the mean/covariance of each component is computed in one step as the weighted mean/covariance of the local cluster of components. The comparison of the time consumptions can be found in Section 5.3.

4. Application to SVM Testing

In this section, we consider how to speed up SVM testing by applying the proposed model simplification scheme. For simplicity, 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

$$f(\mathbf{x}) = \sum_{i=1}^n \beta_i y_i \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2/2h^2) + b, \quad (28)$$

where β_i 's are the Lagrange multipliers, and \mathbf{x}_i 's are the support vectors with corresponding labels y_i 's. Note that (28) is similar in form to a Gaussian mixture model (1), where

3. A spherical bandwidth matrix is of the form $\tilde{\mathbf{H}}_i = \tilde{h}^2 \mathbf{I}$, where \mathbf{I} is the identity matrix.

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 $\alpha_i = y_i \beta_i$. However, some cares should be taken in that (a) the component weights $\alpha_i y_i$'s in (28) can be both positive and negative; (b) the components are un-normalized. For the first issue, we can decompose the model into two sub-models, one including components with positive weights and the other for components with negative weights, and then 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 2.1 can be reused here to reduce the number of components in the SVM decision function, except that a re-scaling 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 for high-dimensional data. So we require the covariances $\tilde{\mathbf{H}}_i$'s of the simplified mixture model (4) 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

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

By using a similar procedure as in Section 2.1, 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

$$\tilde{h}_i^2 = h^2 + \frac{2\tilde{h}_i^2}{h^2 + \tilde{h}_i^2} V_i, \quad (29)$$

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 (29) is a simple scalar equation, which will converge to the following solution

$$\tilde{h}_i^2 = V_i + \sqrt{h^4 + V_i^2}. \quad (30)$$

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

$$w_i = h^d \left(\frac{2\tilde{h}_i^2}{h^2 + \tilde{h}_i^2} \right)^{\frac{d}{2}} \sum_{\mathbf{x}_j \in S_i} \alpha_j \exp \left(-\frac{\|\mathbf{t}_i - \mathbf{x}_j\|^2}{2(h^2 + \tilde{h}_i^2)} \right). \quad (31)$$

Here we have multiplied back the inverse of the normalization factor, $|\mathbf{H}|^{1/2} = h^d$, because the SVM decision function (28) contains un-normalized 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, $i = 1, 2, \dots, m$, 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 (31).

We can also design the spherical covariance version of the method by (Goldberger and Roweis, 2005). There, the local KL-divergence objective (21) can be written as

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

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

$$\sum_{j \in S_i} \alpha_j KL(\phi_j(x) || g_i(x)) = \sum_{j \in S_i} \alpha_j \left(d \frac{h^2}{\tilde{h}_i^2} - 2d \log \left(\frac{h}{\tilde{h}_i} \right) - d + \frac{\|\mathbf{x}_j - \mathbf{t}_i\|^2}{\tilde{h}_i^2} \right). \quad (32)$$

Computing the derivative of (32) 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 (28), 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 (28).

5. Experiments

In this section, we perform experiments to evaluate the proposed mixture simplification scheme. 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 5.1, we compare our method with (Goldberger and Roweis, 2005) (called “GR” in the sequel) on approximating a toy mixture model with varying dimensionality. In Section 5.2, 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 5.3, 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.

5.1 Simplifying Toy Mixture Models

In this section we generate a mixture model whose component centers are 500 d -dimensional points that are uniformly distributed in $[0, 1]$ on each dimension. The weight for each point is also randomly chosen in $[0, 1]$, and the covariance h of the Gaussian component 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 here are the L_2 error, standard KL-divergence, and the local KL-distance defined in (21). Note that the local KL-distance is dependent on the partitioning of the components. Therefore, we use the same partitioning for both methods for a more meaningful comparison.

Results are shown in Figure 1. As can be seen, the errors of both methods grow exponentially with dimensionality. This is because the width h of the Gaussian component is kept constant across the different dimensions. As a result, with growing dimensionality, the points become further apart and each Gaussian component gradually shrinks and ultimately becomes an impulse function in infinitely-high dimensional spaces, making the approximation difficult with the limited number of components. In terms of the L_2 and KL distances, our method is more accurate (on average, ours are 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 (Goldberger and Roweis, 2005).

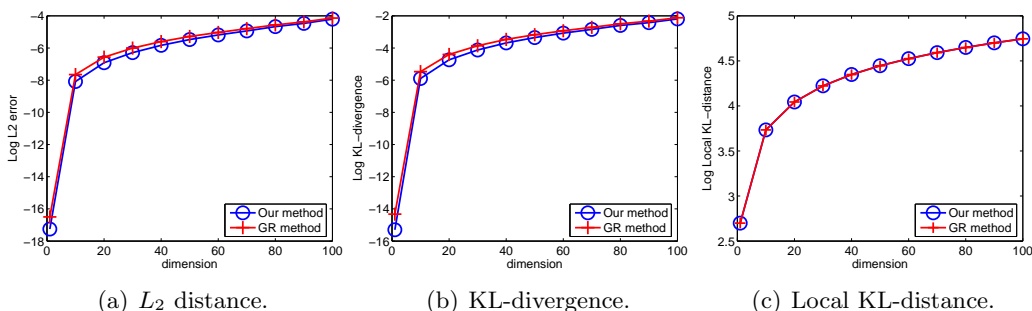


Figure 1: Performance of model simplification with different input dimensionalities.

5.2 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 (Comaniciu and Meer, 2002). As is briefly reviewed in Section 3.3, 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 $O(n^2)$ (Bogdan Georgescu et al., 2003) and can be quite expensive on large samples. 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 some benchmark images that have been extensively tested in the mean shift literature⁴ (Comaniciu and Meer, 2002) (Figure 2). The 3-dimensional 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 . 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). In the experiments, we fix $r = 25$. Segmentation results are shown in Figure 2. The time consumption and the number of components m are shown in Table 1. As can be seen, our segmentation results are very close to those of the standard mean shift algorithm, and the number of components m is significantly smaller than the original component size n used in the Parzen window estimator.

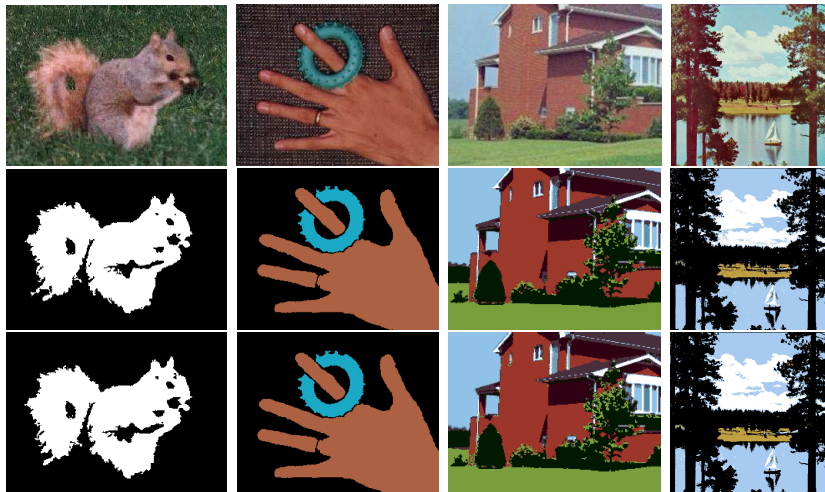


Figure 2: Image segmentation results. Row 1: Original image; Row 2: Standard mean shift; Row 3: Our method.

4. <http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html>

We also compare our approach with the kd-tree-based method in speeding up the mean shift clustering. Note that the mean shift iterations shift the point at \mathbf{x} using (24). In the case of the Gaussian kernel with bandwidth matrix $\mathbf{H} = h^2$, each sample \mathbf{x}_i is assigned a weight $k'(-\|\mathbf{x} - \mathbf{x}_i\|^2/2h^2)$ which can be ignored if the distance $\|\mathbf{x} - \mathbf{x}_i\|$ is larger than some threshold τ (say, $\tau = 3h$). Therefore, in practice, one only needs to search for neighbors of \mathbf{x} such that $\{\mathbf{x}_i : \|\mathbf{x} - \mathbf{x}_i\| \leq \tau\}$. As is mentioned in Section 1, this (spherical) range searching can be done efficiently using spatial data structures like the kd-trees (Samet, 1990) and locality sensitive hashing (Gionis et al., 1999; Bogdan Georgescu et al., 2003). In particular, kd-trees have been used in this manner for accelerating kernel summations in the EM algorithm (Moore, 1998).

We use the kd-tree⁵ for speeding up the neighborhood search in computing the mean shift vector in the standard mean shift. Note that the proposed method can also utilize kd-trees for additional speedup, though this is not implemented in the current version. It is because the number (m) of centers \mathbf{t}_i 's in the simplified model g is already quite small (Table 1). So, instead of using kd-trees, we simply compute the distances between \mathbf{x} and all centers \mathbf{t}_i 's in order to pick out the neighbors that are within a distance of $\tau = 3h$.

Table 1: Time consumption (in seconds) of standard mean shift and its fast version using our method and kd-trees. The numbers of components (m) used in the simplified model are 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)

As can be seen from Table 1, performing mean shift on the original Parzen window density estimator, even with the use of kd-trees, is still significantly slower than the proposed approach. Table 2 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 3,000 as shown in Table 2) 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.

5.3 SVM Testing

In this section, we perform experiments on simplifying the SVM decision function. As discussed in Section 4, we use a spherical covariance for the simplified mixture components. The first data set is the USPS database⁶ for digit recognition. It contains 7,291 training

5. The code is downloaded from the ANN library (<http://www.cs.umd.edu/~mount/ANN>).

6. <ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/>

Table 2: Breakdown of the time consumption (in seconds) by our method and the kd-trees.

image	build model		range searching		clustering		neighborhood size		total time	
	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

samples (16×16 gray image) and 2,007 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 RBF kernel $\exp(-\|x - x'\|^2/\gamma)$. The obtained parameter values are listed in Table 3. Then we simplify the obtained SVM decision function using GR and our method. Note that we first decompose the SVM decision function into two sub-models, one with components having positive $\alpha_i y_i$'s and the other with components having negative $\alpha_i y_i$'s. For each sub-model, 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 3, the actual number of components (m) is usually smaller than 5% of n .

Table 3: The USPS digit data set and the parameters used.

digits	training set	test set	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

Both methods use the same initial partitioning for the same run. Here, the initial partitioning is obtained by the weighted k -means using random seeds on the support vectors, with the multipliers α_i 's as the weights. Each classification task is repeated 100 times. The performance is reported in Tables 4 and 5, where we show the mean (and standard deviation) of the classification error and time consumption. We also perform paired t -test on

the difference between the errors by GR and our method. Here, a positive t -value indicates that GR has a larger error.

Table 4: Classification errors (%) 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 5: Time consumption (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

For more benchmark comparisons, we also experiment with some other data sets from the libsvm data archive⁷ (Table 6). For those data sets with available training and testing splits (`svmguide1a`, `splice`, `w1a`, `adult1a`), we first perform cross-validation on the training data (20% of the training data for validation) and obtain the optimal C and γ parameters. GR and our method are used to simplify the resultant decision function, and their prediction errors on the test data⁸ are computed. We repeat both methods 100 times for each task due

7. <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

8. The test sets are quite large for `w1a` (47,272) and `adult1a` (30,956), so we randomly choose 1000 testing samples.

Table 6: Data sets from the libsvm data archive and the parameters used.

digit	training set	test set	dim	γ	C	(average) #SV (n)	(average) m
svmguide1-a	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
breastcancer	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
liverdisorder	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
ionosphere	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

to the random initialization. For those data sets with only the training part provided, after we use cross-validation to choose the optimal C and γ , we randomly split the data into a training and testing part 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 6. For most data sets, we choose the number of components m as 5% of the number of support vectors. However, for the *ionosphere* and *sonar*, 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. The performance is reported in Table 7, where we show the mean (and standard deviation) of the errors, as well as the t -value of the paired t -test.

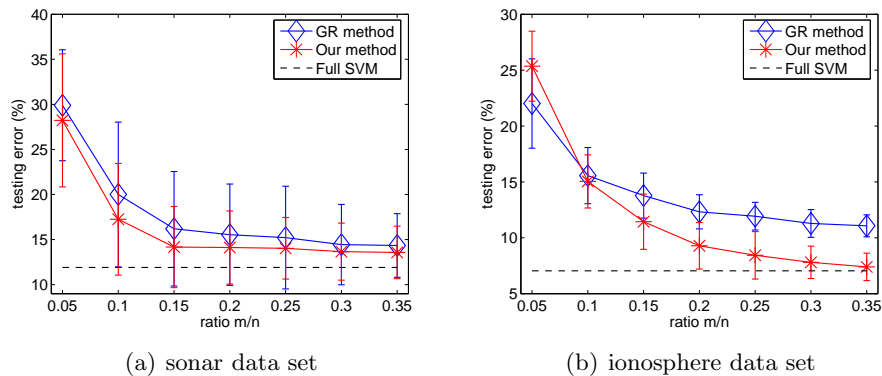


Figure 3: The testing error versus the number of components used in our method and the GR method.

Table 7: Classification errors (%) on the UCI and libsvm data sets.

	original	GR	ours	t (confidence)
svmguidel1a	3.325	4.828±1.752	3.600±0.302	7.299(99.9%)
w1a	2.200	4.412±7.386	2.384±0.348	2.746(99.5%)
splice	10.345	12.371±0.667	11.219±0.592	19.640(99.9%)
adult1a	14.500	14.193±0.520	14.220±0.375	-0.597(—)
german	24.500	26.815±3.145	25.830±2.811	4.268(99.9%)
pima	23.091±2.706	25.318±3.288	24.234±3.043	5.659(99.9%)
breastcancer	2.861±1.239	2.949±1.270	2.920±1.183	0.684(—)
australian	14.986±2.798	14.964±2.810	15.051±2.902	-1.146(—)
heart	17.093±4.535	16.648±4.502	16.685±4.631	-0.332(—)
liver	27.507±5.046	21.855±13.967	21.072±11.927	1.220(—)
diabetes	22.877±2.861	22.864±2.993	22.831±2.856	0.328(—)
ionosphere	5.451±2.441	13.634±4.547	12.859±4.820	1.468(90.0%)
sonar	11.500±4.321	24.929±7.940	20.476±6.985	7.016(99.9%)

Table 8: Time consumption (seconds) on the UCI and libsvm data sets.

	GR	ours
svmguidel1a	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
breastcancer	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
ionosphere	0.007±0.008	0.032±0.012
sonar	0.007±0.008	0.025±0.008

We can see from Tables 4 and 7 that by using only 5% of the support vectors, both methods achieve the testing performance that is close to the original SVM classifier. In terms of the classification error, the difference between the two methods is not statistically significant for the `adult1a`, `breastcancer`, `australian`, `heart`, `liver`, `diabetes` data sets; 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 Tables 5 and 8, the GR method is only faster than our approach by 2 to 4 times.

To examine how the number of components influences the performance of the two algorithms, we also perform more experiments on the two data sets `ionosphere` and `sonar` whose decision functions are relatively difficult to approximate. We split them into fixed training and test sets, and train a SVM using the parameters reported in Table 6. In approximating the decision function, we gradually increase m from 5% to 35% over 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 Figure 3. As can be seen, both methods have improved performance when the number of components m increases.

6. Conclusion

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. Our algorithm can simultaneously solve the problems of *component clustering* and *model simplification*, due to the squared loss function used in measuring model dissimilarities. 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. One is on the optimal number of components in the simplified model. Intuitively, this involves detection of component clusters that will be easily represented by a single kernel. How to perform such detection in an efficient way is thus an interesting problem. Second, our algorithm can be deemed as a pre-image problem that allows variable kernel mapping. It would be interesting to study its performance in tasks like feature extraction and de-noising. Moreover, we are also interested in using a combination of different types of kernels for function approximation. This can be more effective than adopting a single type of kernel.

Appendix A. Proof of Proposition 1

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

$$\mathcal{Y}_i = \int \left(\mathcal{R}_i(\mathbf{x}) - \frac{1}{Z_i} \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x}, \quad (33)$$

and $Z_i = \sum_{j \in S_i} \alpha_j$. Note that both \mathcal{Q} (11) and \mathcal{Y} (33) are broken into m independent terms under the use of the L_2 distance measure. 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 in the same functional form as (3).

Note that \mathcal{Q}_i can be written as

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

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

$$Z_i \mathcal{Y}_i(\mathcal{R}_i) = Z_i \int \mathcal{R}_i(\mathbf{x})^2 d\mathbf{x} + \frac{1}{Z_i} \int \left(\sum_{j \in \mathcal{S}_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x} - 2 \sum_{j \in \mathcal{S}_i} \int \alpha_j \phi_j(\mathbf{x}) \mathcal{R}_i(\mathbf{x}) d\mathbf{x}, \quad (35)$$

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

From (34) and (35), 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 . Note further from the definitions of $\bar{\mathcal{E}}_i$ in (8) and \mathcal{Y}_i in (33) that their optimal solutions are related by $w_i g_i(\mathbf{x}) = Z_i \mathcal{R}_i^{\mathcal{Y}}(\mathbf{x})$. So, we have $w_i g_i(\mathbf{x}) = Z_i \mathcal{R}_i^{\mathcal{Q}}(\mathbf{x})$. Plugging this into (8), we have

$$\begin{aligned}\bar{\mathcal{E}} &= m \sum_{i=1}^m \int \left(Z_i \mathcal{R}_i^{\mathcal{Q}}(\mathbf{x}) - \sum_{j \in \mathcal{S}_i} \alpha_j \phi_j(\mathbf{x}) \right)^2 d\mathbf{x} \\ &= m \sum_{i=1}^m \int \left(\sum_{j \in \mathcal{S}_i} \alpha_j (\mathcal{R}_i^{\mathcal{Q}}(\mathbf{x}) - \phi_j(\mathbf{x})) \right)^2 d\mathbf{x} \\ &\leq m \bar{n} \sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \alpha_j^2 \int (\mathcal{R}_i^{\mathcal{Q}}(\mathbf{x}) - \phi_j(\mathbf{x}))^2 d\mathbf{x} \equiv \tilde{\mathcal{E}}, \quad (36)\end{aligned}$$

by using the inequality in (7). Here, \bar{n} is the largest cluster size. During the iterations of component clustering in (10), each component ϕ_j is re-assigned to its closest representative $\mathcal{R}_{\pi(j)}$ where $\pi(j) = \arg \min_{i=1, \dots, m} \int (\phi_j(\mathbf{x}) - \mathcal{R}_i(\mathbf{x}))^2 d\mathbf{x}$. Such a re-assignment can also reduce $\tilde{\mathcal{E}}$ in (36). Hence, the component clustering procedure can find a local minimum of the model-simplification error's upper bound $\bar{\mathcal{E}}$.

As for the local approximation step, since it is performing coordinate descent on $\bar{\mathcal{E}}$, it can find a local minimum of $\bar{\mathcal{E}}$ and, as we have proved, $\bar{\mathcal{E}} \leq \tilde{\mathcal{E}}$.

Appendix B. Derivation of the Local Model-Simplification Error $\bar{\mathcal{E}}_i$ (15)

We first introduce two lemmas.

Lemma 1 For $\mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^{d \times 1}$, and non-singular $\mathbf{H} \in \mathbb{R}^{d \times d}$, the following holds

$$\int |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} = C_{Kd},$$

where C_{Kd} is a constant depending 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

$$\begin{aligned} \int |\mathbf{H}|^{-1/2} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} &= |\mathbf{H}|^{-1/2} \int K\left(\mathbf{H}^{-1/2}(\mathbf{x} - \mathbf{x}_0)\right) d\mathbf{x} \\ &= |\mathbf{H}|^{-1/2} \int K(\mathbf{y}) |\mathbf{H}|^{1/2} d\mathbf{y} \\ &= \int K(\mathbf{y}) d\mathbf{y}, \end{aligned}$$

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 non-singular $\mathbf{H}_1, \mathbf{H}_2 \in \mathbb{R}^{d \times d}$, the following holds

$$(\mathbf{x} - \mathbf{x}_1)^\top \mathbf{H}_1^{-1} (\mathbf{x} - \mathbf{x}_1) + (\mathbf{x} - \mathbf{x}_2)^\top \mathbf{H}_2^{-1} (\mathbf{x} - \mathbf{x}_2) = (\mathbf{x} - \mathbf{z})^\top \mathbf{H}^{-1} (\mathbf{x} - \mathbf{z}) + r, \quad (37)$$

where

$$\mathbf{H} = (\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1})^{-1}, \quad (38)$$

$$\mathbf{z} = \mathbf{H} (\mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{H}_2^{-1} \mathbf{x}_2), \quad (39)$$

$$r = (\mathbf{x}_1 - \mathbf{x}_2)^\top (\mathbf{H}_1 + \mathbf{H}_2)^{-1} (\mathbf{x}_1 - \mathbf{x}_2). \quad (40)$$

Proof The left hand side of (37) can be written as

$$\mathbf{x}^\top (\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}) \mathbf{x} - 2(\mathbf{x}^\top \mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{x}^\top \mathbf{H}_2^{-1} \mathbf{x}_2) + \mathbf{x}_1^\top \mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{x}_2^\top \mathbf{H}_2^{-1} \mathbf{x}_2. \quad (41)$$

The right hand side of (37) can be written as

$$\begin{aligned} &\mathbf{x}^\top \mathbf{H}^{-1} \mathbf{x} - 2\mathbf{x}^\top \mathbf{H}^{-1} \mathbf{z} + \mathbf{z}^\top \mathbf{H}^{-1} \mathbf{z} + r \\ &= \mathbf{x}^\top (\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}) \mathbf{x} - 2(\mathbf{x}^\top \mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{x}^\top \mathbf{H}_2^{-1} \mathbf{x}_2) + \mathbf{z}^\top \mathbf{H}^{-1} \mathbf{z} + r, \end{aligned} \quad (42)$$

where we have used the simple equality $\mathbf{H}^{-1} = \mathbf{H}_1^{-1} + \mathbf{H}_2^{-1}$ by (38) and $\mathbf{H}^{-1} \mathbf{z} = \mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{H}_2^{-1} \mathbf{x}_2$ by (39). Now, compare (41) and (42), we only need to prove

$$\mathbf{x}_1^\top \mathbf{H}_1^{-1} \mathbf{x}_1 + \mathbf{x}_2^\top \mathbf{H}_2^{-1} \mathbf{x}_2 = \mathbf{z}^\top \mathbf{H}^{-1} \mathbf{z} + r. \quad (43)$$

We start from the right hand side of equation (43)

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

By using the equality $(\mathbf{H}_1^{-1} + \mathbf{H}_2^{-1})^{-1} = \mathbf{H}_1(\mathbf{H}_1 + \mathbf{H}_2)^{-1}\mathbf{H}_2 = \mathbf{H}_2(\mathbf{H}_1 + \mathbf{H}_2)^{-1}\mathbf{H}_1$,

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

So (43) holds and Lemma 2 is proved. \blacksquare

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

$$\begin{aligned}
 K_{\mathbf{H}_1}(\mathbf{x} - \mathbf{x}_1) K_{\mathbf{H}_2}(\mathbf{x} - \mathbf{x}_2) &= k \left((\mathbf{x} - \mathbf{x}_1)^\top \mathbf{H}_1^{-1} (\mathbf{x} - \mathbf{x}_1) + (\mathbf{x} - \mathbf{x}_2)^\top \mathbf{H}_2^{-1} (\mathbf{x} - \mathbf{x}_2) \right) \\
 &= k(r) K_{\mathbf{H}}(\mathbf{x} - \mathbf{z}),
 \end{aligned} \tag{44}$$

with \mathbf{H} , \mathbf{z} and r given in (38), (39) and (40), respectively.

Now we begin to derive $\bar{\mathcal{E}}_i$ (15). Using (8), we have

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

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

$$\begin{aligned}
 \int w_i^2 g_i^2(\mathbf{x}) d\mathbf{x} &= \int w_i^2 C_{Kd}^{-2} |\tilde{\mathbf{H}}_i|^{-1} K_{\tilde{\mathbf{H}}_i/2}(\mathbf{x} - \mathbf{t}_i) d\mathbf{x} \quad (\text{using (5)}) \\
 &= w_i^2 C_{Kd}^{-1} |2\tilde{\mathbf{H}}_i|^{-1/2} \int C_{Kd}^{-1} |\tilde{\mathbf{H}}_i/2|^{-1/2} K_{\tilde{\mathbf{H}}_i/2}(\mathbf{x} - \mathbf{t}_i) d\mathbf{x} \\
 &= w_i^2 C_{Kd}^{-1} |2\tilde{\mathbf{H}}_i|^{-1/2}. \quad (\text{using Lemma 1})
 \end{aligned}$$

For the third term,

$$\begin{aligned}
 2 \int w_i g_i(\mathbf{x}) \sum_{j \in \mathcal{S}_i} \alpha_j \phi_j(\mathbf{x}) d\mathbf{x} &= \sum_{j \in \mathcal{S}_i} \int \frac{2w_i \alpha_j C_{Kd}^{-2}}{|\tilde{\mathbf{H}}_i \mathbf{H}_j|^{1/2}} K_{\tilde{\mathbf{H}}_i}(\mathbf{x} - \mathbf{t}_i) K_{\mathbf{H}_j}(\mathbf{x} - \mathbf{x}_j) d\mathbf{x} \\
 &\quad (\text{using (3) and (5)}) \\
 &= \sum_{j \in \mathcal{S}_i} \frac{2w_i \alpha_j k(r_{ij}) C_{Kd}^{-1}}{|\tilde{\mathbf{H}}_i \mathbf{H}_j|^{1/2}} \int C_{Kd}^{-1} K_{\mathbf{H}_{ij}}(\mathbf{x} - \mathbf{z}_{ij}) d\mathbf{x}, \\
 &\quad (\text{using (44) and Lemma 2})
 \end{aligned}$$

where

$$\begin{aligned}
 \mathbf{H}_{ij} &= \left(\tilde{\mathbf{H}}_i^{-1} + \mathbf{H}_j^{-1} \right)^{-1}, \\
 r_{ij} &= (\mathbf{x}_j - \mathbf{t}_i)^\top (\tilde{\mathbf{H}}_i + \mathbf{H}_j)^{-1} (\mathbf{x}_j - \mathbf{t}_i), \\
 \mathbf{z}_{ij} &= \left(\tilde{\mathbf{H}}_i^{-1} + \mathbf{H}_j^{-1} \right)^{-1} \left(\tilde{\mathbf{H}}_i^{-1} \mathbf{t}_i + \mathbf{H}_j^{-1} \mathbf{x}_j \right).
 \end{aligned} \tag{45}$$

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

$$\begin{aligned} 2 \int w_i g_i(\mathbf{x}) \sum_{j \in S_i} \alpha_j \phi_j(\mathbf{x}) d\mathbf{x} &= \sum_{j \in S_i} \frac{2w_i \alpha_j k(r_{ij}) C_{Kd}^{-1}}{|\tilde{\mathbf{H}}_i \mathbf{H}_j|^{1/2}} |\mathbf{H}_{ij}|^{1/2} \\ &= 2w_i \sum_{j \in S_i} \frac{\alpha_j C_{Kd}^{-1} k(r_{ij})}{|\tilde{\mathbf{H}}_i + \mathbf{H}_j|^{1/2}}. \quad (\text{using (45)}) \end{aligned}$$

By combining all three terms, we obtain (15). Next we show how to compute $D(\phi_j, \mathcal{R}_i)$ (line 22 in Algorithm 1). Note that $D(\phi_j, \mathcal{R}_i) = D(\frac{w_i g_i}{\mathbf{Z}_i}, \phi_j)$ according to Proposition 1, while equation (15) is equal to $D(w_i g_i, \sum_{j \in S_i} \alpha_j \phi_j)$. 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) $w_i \leftarrow w_i/\mathbf{Z}_i$; (2) $j \leftarrow j \in S_i$; (3) $\alpha_j \leftarrow 1$.

References

- G. A. Babich and O. I. Camps. Weighted Parzen windows for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):567–570, 1996.
- A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.
- J. C. Bezdek and R. J. Hathaway. Convergence of alternating optimization. *Neural, Parallel & Scientific Computations*, 11(4):351–368, 2003.
- B. Bogdan Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimenisons, a texture classification example. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, pages 456–463, 2003.
- C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1996. MIT Press.
- D. Comaniciu. An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:281–288, 2003.
- D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1197–1204, 1999.
- D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.
- D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.
- D. D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *IEEE International Conference on Computer Vision*, volume 1, pages 438–445, 2001.

- J. V. Davis and I. Dhillon. Differential entropic clustering of multivariate Gaussians. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.
- D. DeMenthon and D. Doermann. Video retrieval using spatio-temporal descriptors. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, pages 508–517, Berkeley, CA, US, 2003.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977.
- J. Fan and J. S. Marron. Fast implementations of nonparametric curve estimators. *Journal of Computational and Graphical Statistics*, 3(1):35–56, March 1994.
- K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21:32–40, 1975.
- A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, Boston, 1992.
- A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.
- M. Girolami and C. He. Probability density estimation from optimally condensed data samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1253–1264, 2003.
- J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 505–512, Cambridge, MA, 2005. MIT Press.
- B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Incremental density approximation and kernel-based Bayesian filtering for object tracking. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 638–644, 2004.
- W. Härdle and D. W. Scott. Smoothing by weighted averaging of rounded points. *Computational Statistics*, 7:97–128, 1992.
- A. J. Izenman. Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, 86(413):205–224, 1991.
- B. Jeon and D. A. Landgrebe. Fast Parzen density estimation using clustering based branch and bound. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):950–954, 1994.
- 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 Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

- 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 Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1631–1639, December 2003.
- J. S. Marron and M. P. Wand. Exact mean integrated squared error. *Annals of Statistics*, 20(2):712–736, 1992.
- G. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, New York, 1992.
- G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000.
- A. W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 543–549, San Mateo, CA, 1998. Morgan Kaufmann.
- E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1075, 1962.
- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.
- L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.
- K. Roeder. Density estimation with confidence sets exemplified by superclusters and voids in the galaxies. *Journal of the American Statistical Association*, 85(411):617–624, 1990.
- H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, Boston, MA, 1990.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- D. W. Scott and S. J. Sheather. Kernel density estimation with binned data. *Communications in Statistics, Part A – Theory and Methods*, 14:1353 – 1359, 1985.
- E. B. Sudderth, A. T. Ihler, W. T. Freeman, and Willsky A. S. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *Advances in Neural Information Processing Systems 17*, pages 1369–1376, Cambridge, MA, 2005. MIT Press.
- E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 605–612, 2003.
- P. Vincent and Y. Bengio. Manifold Parzen windows. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 825–832, Cambridge, MA, 2003. MIT Press.

- K. Zhang and J. T. Kwok. Simplifying mixture models through function approximation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 17*, pages 1577–1584, Cambridge, MA, 2007. MIT Press.
- X. S. Zhou, D. Comaniciu, and A. Gupta. An information fusion framework for robust shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1): 115–129, January 2005.