Kernel Trick Embedded Gaussian Mixture Model

Jingdong Wang, Jianguo Lee, and Changshui Zhang

State Key Laboratory of Intelligent Technology and Systems Department of Automation, Tsinghua University Beijing, 100084, P. R. China
{wangjd01,lijg01}@mails.tsinghua.edu.cn
zcs@mail.tsinghua.edu.cn

Abstract. In this paper, we present a kernel trick embedded Gaussian Mixture Model (GMM), called kernel GMM. The basic idea is to embed kernel trick into EM algorithm and deduce a parameter estimation algorithm for GMM in feature space. Kernel GMM could be viewed as a Bayesian Kernel Method. Compared with most classical kernel methods, the proposed method can solve problems in probabilistic framework. Moreover, it can tackle nonlinear problems better than the traditional GMM. To avoid great computational cost problem existing in most kernel methods upon large scale data set, we also employ a Monte Carlo sampling technique to speed up kernel GMM so that it is more practical and efficient. Experimental results on synthetic and real-world data set demonstrate that the proposed approach has satisfing performance.

1 Introduction

Kernel trick is an efficient method for nonlinear data analysis early used by Support Vector Machine (SVM) [18]. It has been pointed out that kernel trick could be used to develop nonlinear generalization of any algorithm that could be cast in the term of dot products. In recent years, kernel trick has been successfully introduced into various machine learning algorithms, such as Kernel Principal Component Analysis (Kernel PCA) [14,15], Kernel Fisher Discriminant (KFD) [11], Kernel Independent Component Analysis (Kernel ICA) [7] and so on.

However, in many cases, we are required to obtain risk minimization result and incorporate prior knowledge, which could be easily provided within Bayesian probabilistic framework. This makes the emerging of combining kernel trick and Bayesian method, which is called Bayesian Kernel Method [16]. As Bayesian Kernel Method is in probabilistic framework, it can realize Bayesian optimal decision and estimate confidence or reliability easily with probabilistic criteria such as *Maximum-A-Posterior* [5] and so on.

Recently some researches have been done in this field. Kwok combined the evidence framework with SVM [10], Gestel et al. [8] incorporated Bayesian framework with SVM and KFD. These two work are both to apply Bayesian framework to known kernel method. On the other hand, some researchers proposed

R. Gavaldà et al. (Eds.): ALT 2003, LNAI 2842, pp. 159–174, 2003.

new Bayesian methods with kernel trick embedded, among which one of the most influential work is the Relevance Vector Machine (RVM) proposed by Tipping [17].

This paper also addresses the problem of Bayesian Kernel Method. The proposed method is that we embed kernel trick into Expectation-Maximization (EM) algorithm [3], and deduce a new parameter estimation algorithm for Gaussian Mixture Model (GMM) in the feature space. The entire model is called kernel Gaussian Mixture Model (kGMM).

The rest of this paper is organized as follows. Section 2 reviews some background knowledge, and Section 3 describes the kernel Gaussian Mixture Model and the corresponding parameter estimation algorithm. Experiments and results are presented in Section 4. Conclusions are drawn in the final section.

2 Preliminaries

In this section, we review some background knowledge including the kernel trick, GMM based on EM algorithm and Bayesian Kernel Method.

2.1 Kernel Trick

Mercer kernel trick was early applied by SVM. The idea is that we can implicitly map input data into a high dimension feature space via a nonlinear function:

$$\begin{aligned} \Phi : X &\to H \\ x &\mapsto \phi(x) \end{aligned} \tag{1}$$

And a similarity measure is defined from the dot product in space H as follows:

$$k\left(x, x'\right) \triangleq \left(\phi(x) \cdot \phi(x')\right) \tag{2}$$

where the kernel function k should satisfy *Mercer's condition* [18]. Then it allows us to deal with learning algorithms using linear algebra and analytic geometry.

Generally speaking, on the one hand kernel trick could deal with data in the high-dimensional dot product space H, which is named feature space by a map associated with k. On the other hand, it avoids expensive computation cost in feature space by employing the kernel function k instead of directly computing dot product in H.

Being an elegant way for nonlinear analysis, kernel trick has been used in many other algorithms such as Kernel Fisher Discriminant [11], Kernel PCA [14, 15], Kernel ICA [7] and so on.

2.2 GMM Based on EM Algorithm

GMM is a kind of mixture density models, which assumes that each component of the probabilistic model is a Gaussian density. That is to say:

$$p(x|\Theta) = \sum_{i=1}^{M} \alpha_i G_i(x|\theta_i) \tag{3}$$

where $x \in \mathbf{R}^d$ is a random variable, parameters $\Theta = \left(\alpha_1, \cdots, \alpha_M; \theta_1, \cdots \theta_M\right)$ satisfy $\sum_{i=1}^{M} \alpha_i = 1$, $\alpha_i \geq 0$ and $G_i(x|\theta_i)$ is a Gaussian probability density function:

$$G_l(x|\theta_l) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_l\right|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l)\right\} \tag{4}$$

where $\theta_l = (\mu_l, \Sigma_l)$.

GMM could be viewed as a generative model [12] or a latent variable model [6] that assumes data set $X = \{x_i\}_{i=1}^{N}$ are generated by M Gaussian components, and introduces the latent variables item $Z = \{z_i\}_{i=1}^{N}$ whose value indicates which component generates the data. That is to say, we assume that if sample x_i is generated by the l^{th} component and then $z_i = l$. Then the parameters of GMM could be estimated by the EM algorithm [2].

EM algorithm for GMM is an iterative procedure, which estimates the new parameters in terms of the old parameters as the following updating formulas:

$$\alpha_l^{(t)} = \frac{1}{N} \sum_{i=1}^{N} p(l|x_i, \Theta^{(t-1)})$$

$$\mu_l^{(t)} = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^{(t-1)})}{\sum_{i=1}^{N} p(l|x_i, \Theta^{(t-1)})}$$

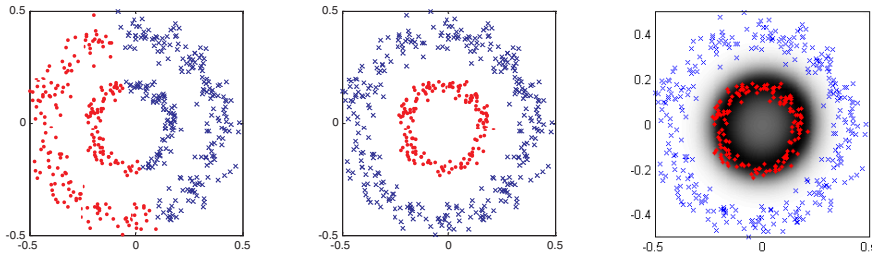
$$\Sigma_l^{(t)} = \frac{\sum_{i=1}^{N} (x_i - \mu_l^{(t)})(x_i - \mu_l^{(t)})^T p(l|x_i, \Theta^{(t-1)})}{\sum_{i=1}^{N} p(l|x_i, \Theta^{(t-1)})} \tag{5}$$

where l represents the l^{th} Gaussian component, and

$$p(l|x_i, \Theta^{(t-1)}) = \frac{\alpha_l^{(t-1)} G(x_i|\theta_l^{(t-1)})}{\sum_{j=1}^{M} \alpha_j^{(t-1)} G(x_i|\theta_j^{(t-1)})}, \ l = 1, \cdots, M \tag{6}$$

$\Theta^{(t-1)} = \left(\alpha_1^{(t-1)}, \cdots, \alpha_M^{(t-1)}; \theta_1^{(t-1)}, \cdots, \theta_M^{(t-1)}\right)$ are parameters of the $(t-1)^{th}$ iteration and $\Theta^{(t)} = \left(\alpha_1^{(t)}, \cdots, \alpha_M^{(t)}; \theta_1^{(t)}, \cdots, \theta_M^{(t)}\right)$ are parameters of the $(t)^{th}$ iteration.

GMM has been successfully applied in many fields, such as parametric clustering, density estimation and so on. However, for instance, it can't give a simple but satisfied clustering result on data set with complex structure [13] as shown in Figure 1. One alternative way is to perform GMM based clustering in another space instead of in original data space.



(a) (b) (c)

Fig. 1. Data set of two concentric circles with 1,000 samples, points marked by '×' belong to one cluster and marked by '·' belong to the other. (a) is the partition result by traditional GMM, (b) is the result achieved by kGMM using polynomial kernel of degree 2; (c) shows the probability that each point belongs to the outer circle. The whiter the point is, the higher the probability is.

2.3 Bayesian Kernel Method

Bayesian Kernel Method could be viewed as a combination of Bayesian method and Kernel method. It inherits merits from both these two methods. It could tackle problems nonlinearly like kernel method, and it obtains estimation results within a probabilistic framework like classical Bayesian methods. Many works have been done in this field.

There are typically three different ways.

- Interpretation of kernel methods in Bayesian framework such as SVM and other kernel methods in Bayesian framework as in [10,8];
- Employing kernel methods in traditional Bayesian methods such as Gaussian Processes and Laplacian Processes [16];
- Proposing new methods in Bayesian framework with kernel trick embedded such as Relevance Vector Machine (RVM) [17] and Bayes Point Machine [9].

And we intend to embed kernel trick into Gaussian Mixture Model. This work just belongs to the second category of Bayesian Kernel Method.

3 Kernel Trick Embedded GMM

As mentioned before, Gaussian Mixture Model can not obtain simple but satisfied results on data sets with complex structure, so we consider employing kernel trick to realize a Bayesian Kernel version of GMM. Our basic idea is to embed kernel trick into parameter estimation procedure of GMM. In this section, we firstly describe GMM in feature space, secondly present the properties in feature space, then formulate the Kernel Gaussian Mixture Model and the corresponding parameter estimation algorithm, finally make some discussions on the algorithm.

3.1 GMM in Feature Space

GMM in feature space by a map $\phi(\cdot)$ associated with kernel function k can be easily rewritten as

$$p(\phi(x)|\Theta) = \sum_{i=1}^{M} \alpha_i G(\phi(x)|\theta_i) \tag{7}$$

and the EM updating formula in (5) and (6) can be replaced by the following.

$$\alpha_l^{(t)} = \frac{1}{N} \sum_{i=1}^{N} p(l|\phi(x_i), \Theta^{(t-1)})$$

$$\mu_l^{(t)} = \frac{\sum_{i=1}^{N} \phi(x_i) p(l|\phi(x_i), \Theta^{(t-1)})}{\sum_{i=1}^{N} p(l|\phi(x_i), \Theta^{(t-1)})}$$

$$\Sigma_l^{(t)} = \frac{\sum_{i=1}^{N} (\phi(x_i) - \mu_l^{(t)})(\phi(x_i) - \mu_l^{(t)})^T p(l|\phi(x_i), \Theta^{(t-1)})}{\sum_{i=1}^{N} p(l|\phi(x_i), \Theta^{(t-1)})} \tag{8}$$

where l represents the l^{th} Gaussian component, and

$$p(l|\phi(x_i), \Theta^{(t-1)}) = \frac{\alpha_l^{(t-1)} G(\phi(x_i)|\theta_l^{(t-1)})}{\sum_{j=1}^{M} \alpha_j^{(t-1)} G(\phi(x_i)|\theta_j^{(t-1)})}, l = 1, \cdots, M \tag{9}$$

However, computing GMM directly with formula (8) and (9) in a high dimension feature space is computationally expensive thus impractical. We consider employing kernel trick to overcome this difficulty. In the following section, we will give some properties based on *Mercer kernel trick* to estimate the GMM parameters in feature space.

3.2 Properties in Feature Space

To be convenient, notations in feature space are given firstly, and then three properties are presented.

Notations. In all the formulas, bold and capital letters are for matrixes, italic and bold letters are for vectors, and italic and lower case are for scalars. Subscript l represents the l^{th} Gaussian component, superscript t represents the t^{th} iteration of the EM procedure. A^T represents transpose of matrix A. Other notations are shown in Table 1.

Table 1. Notation List

$p_{li}^{(t)} = p(l\|\phi(x_i), \Theta^{(t)})$	Posterior that $\phi(x_i)$ belongs to the l^{th} component.
$w_{li}^{(t)} = \sqrt{(p_{li}^{(t)} / \sum_{j=1}^{M} p_{ji}^{(t)})}$	$(w_{li}^{(t)})^2$ represents ratio that $\phi(x_i)$ is occupied by the l^{th} Gaussian component.
$\mu_l^{(t)} = \sum_{i=1}^{N} \phi(x_i)(w_{li}^{(t)})^2$	Mean vector of the l^{th} Gaussian component.
$\tilde{\phi}_l^{(t)}(x_i) = \phi(x_i) - \mu_l^{(t)}$	Centered image of $\phi(x_i)$.
$\Sigma_l^{(t)} = \sum_{i=1}^{N} \tilde{\phi}(x_i)\tilde{\phi}(x_i)^T (w_{li}^{(t)})^2$	Covariance matrix of the l^{th} Gaussian component.
$(\mathbf{K}_l^{(t)})_{ij} = \big((w_{li}\phi(x_i) \cdot w_{lj}\phi(x_j)\big)$	Kernel matrix
$(\tilde{\mathbf{K}}_l^{(t)})_{ij} = \big((w_{li}\tilde{\phi}(x_i) \cdot w_{lj}\tilde{\phi}(x_j)\big)$	Centered kernel matrix.
$(\mathbf{K}_l^{(t)})'_{ij} = \big((\phi(x_i) \cdot w_{lj}\phi(x_j)\big)$	Projecting kernel matrix.
$(\tilde{\mathbf{K}}_l^{(t)})'_{ij} = \big(\tilde{\phi}(x_i) \cdot w_{lj}\tilde{\phi}(x_j)\big)$	Centered projecting kernel matrix.
$\lambda_{le}^{(t)},\ V_{le}^{(t)}$	Eigenvalue and eigenvector of $\Sigma_l^{(t)}$.
$\lambda_{le}^{(t)},\ \beta_{le}^{(t)}$	Eigenvalue and eigenvector of $\tilde{\mathbf{K}}_l^{(t)}$.

Properties in Feature Space. According to Lemma 1 in Appendix, we can get the first property.

[**Property 1**] Centered kernel matrix $\tilde{\mathbf{K}}_l^{(t)}$ and centered projecting kernel matrix $(\tilde{\mathbf{K}}_l^{(t)})'$ are computed from the following formulas:

$$\tilde{\mathbf{K}}_l^{(t)} = \mathbf{K}_l^{(t)} - \mathbf{W}_l^{(t)}\mathbf{K}_l^{(t)} - \mathbf{K}_l^{(t)}\mathbf{W}_l^{(t)} + \mathbf{W}_l^{(t)}\mathbf{K}_l^{(t)}\mathbf{W}_l^{(t)}$$

$$(\tilde{\mathbf{K}}_l^{(t)})' = (\mathbf{K}_l^{(t)})' - (\mathbf{W}_l^{(t)})'\mathbf{K}_l^{(t)} - (\mathbf{K}_l^{(t)})'\mathbf{W}_l^{(t)} + (\mathbf{W}_l^{(t)})'\mathbf{K}_l^{(t)}\mathbf{W}_l^{(t)} \tag{10}$$

where $\mathbf{W}_l^{(t)} = w_l^{(t)}(w_l^{(t)})^T$, $(\mathbf{W}_l^{(t)})' = 1_N(w_l^{(t)})^T$, $w_l^{(t)} = [w_{l1}^{(t)}, \cdots, w_{lN}^{(t)}]^T$, and 1_N is a N-dimensional column vector with all entries equal to 1.

This property presents the way to center the kernel matrix and the projecting kernel matrix. According to Lemma 2 in Appendix and Property 1, we can obtain the second property.

[**Property 2**] Feature space covariance matrix $\Sigma_l^{(t)}$ and centered kernel matrix $\tilde{\mathbf{K}}_l^{(t)}$ have the same nonzero eigenvalues, and the following equivalence relation holds.

$$\Sigma_l^{(t)} V_{le}^{(t)} = \lambda_{le} V_{le}^{(t)} \Leftrightarrow \tilde{\mathbf{K}}_l^{(t)} \beta_{le}^{(t)} = \lambda_{le} \beta_{le}^{(t)} \tag{11}$$

where $V_{le}^{(t)}$ and $\beta_{le}^{(t)}$ are eigenvectors of $\Sigma_l^{(t)}$ and $\tilde{\mathbf{K}}_l^{(t)}$ respectively.

This property enables us to compute eigenvectors from centered kernel matrix instead of from feature space covariance matrix as in Equation (8).

With the first two properties, we do not need to compute the means and covariance matrixes in feature space, but do the eigen-decomposition of centered kernel matrix instead.

Feature space has a very high dimensionality, which makes it intractable to compute the complete Gaussian probability density $G_l(\phi(x_j)|\theta_l)$ directly. However, Moghadam and Pentland's work [22] bring us some motivation. It divides the original high dimension space into two parts: the principal subspace and the orthogonal complement subspace. The principal subspace has dimension d_ϕ. Thus the complete Gaussian density could be approximated by

$$\hat{G}_l(\phi(x_j)|\theta_l) = \frac{1}{(2\pi)^{d_\phi/2}\prod_{e=1}^{d_\phi}\lambda_{le}^{1/2}}\exp\Big[-\frac{1}{2}\sum_{e=1}^{d_\phi}\frac{y_e^2}{\lambda_{le}}\Big] \times \frac{1}{(2\pi\rho)^{(N-d_\phi)/2}}\exp\Big[-\frac{\varepsilon^2(x_j)}{2\rho}\Big] \tag{12}$$

where $y_e = \tilde{\phi}(x_j)^T V_{le}$ (here V_{le} is the *e-th* eigenvector of Σ_l), ρ is the weight ratio, and $\varepsilon^2(x_j)$ is the residual reconstruction error.

At the right side of the Equation (12), the first factor computes from the principal subspace, and the second factor computes from the orthogonal complement subspace.

The optimal value of ρ can be determined by minimizing a cost function. From an information-theoretic point of view, the cost function should be the Kullback-Leibler divergence between the true density $G_l(\phi(x_j)|\theta_l)$ and its approximation $\hat{G}_l(\phi(x_j)|\theta_l)$.

$$J(\rho) = \int G(\phi(x)|\theta_l)\log\frac{G(\phi(x)|\theta_l)}{\hat{G}(\phi(x)|\theta_l)}\mathrm{d}\phi(x) \tag{13}$$

Plugging (12) into upper equation, it can easily shown that

$$J(\rho) = \frac{1}{2}\sum_{e=d_\phi+1}^{N}\Big[\frac{\lambda_{le}}{\rho} - 1 + \log\frac{\rho}{\lambda_{le}}\Big]$$

Solving the equation $\frac{\partial J}{\partial \rho} = 0$ yields the optimal value

$$\rho* = \frac{1}{N-d_\phi}\sum_{e=d_\phi+1}^{N}\lambda_{le}$$

And according to Property 2, Σ_l and $\tilde{\mathbf{K}}_l$ has same nonzero eigenvalues, by employing the property of symmetry matrix, we obtain

$$\rho* = \frac{1}{N-d_\phi}\Big[\big\|\Sigma_l\big\|_F^2 - \sum_{e=1}^{d_\phi}\lambda_{le}\Big] = \frac{1}{N-d_\phi}\Big[\big\|\tilde{\mathbf{K}}_l\big\|_F^2 - \sum_{e=1}^{d_\phi}\lambda_{le}\Big] \tag{14}$$

where $\|\cdot\|_F$ is a Frobenius matrix norm defined as $\|A\|_F = \sqrt{trace(AA^T)}$.

The residual reconstruction error, $\varepsilon^2(x_j) = \|\tilde{\phi}(x_j)\|^2 - \sum_{e=1}^{d_\phi} y_e^2$, could be easily obtained by employing kernel trick

$$\varepsilon^2(x_j) = k(x_j, x_j) - \sum_{e=1}^{d_\phi} y_e^2 \tag{15}$$

And according to Lemma 3 in Appendix,

$$y_e = \tilde{\phi}(x_j)^T V_{le} = \left(V_{le} \cdot \tilde{\phi}(x_j)\right) = \beta^T \Gamma_l \tag{16}$$

where Γ_l is the *j-th* column of centered projecting kernel matrix $\tilde{\mathbf{K}}'_l$.

It should notice here the centered kernel matrix $\tilde{\mathbf{K}}_l$ is used to obtain eigenvalues λ_l and eigenvectors β_l, whereas projecting kernel matrix $\tilde{\mathbf{K}}'_l$ is used to compute y_e as in Equation (16). And both these two matrixes cannot be omitted in the training procedure.

Employing all these results, we obtain the third property.

[Property 3] The Gaussian probability density function $G_l(\phi(x_j)|\theta_l)$ can be approximated by $\hat{G}_l(\phi(x_j)|\theta_l)$ as shown in (12), where ρ, $\varepsilon^2(x_j)$ and y_e are shown in (14), (15) and (16) respectively.

We should specially stress that the approximation of $G_l(\phi(x_j)|\theta_l)$ by $\hat{G}_l(\phi(x_j)|\theta_l)$ is complete since it represents not only the principle subspace but also the orthogonal complement subspace.

According to these three properties, we can draw the following conclusions.

- Mercer kernel trick is introduced to indirectly compute dot products in the high dimension feature space.
- The probability that a sample belongs to the l^{th} component can be computed through centered kernel matrix and centered projecting kernel matrix instead of mean vector and covariance matrix as in traditional GMM.
- Needing not to obtain full eigen-decomposition of the centered kernel matrix, we could approximate the Gaussian density with only the largest d_ϕ principal components of the centered kernel matrix, and the approximation is not dependent on d_ϕ very much since it is complete and optimal.

With these three properties, we can formulate our kernel Gaussian Mixture Model.

3.3 Kernel GMM and the Parameter Estimation Algorithm

In feature space, the kernel matrixes replace the mean and covariance matrixes in input space to represent the Gaussian component. So the parameters of each component are not the mean vectors and covariance matrixes, but the kernel matrixes. In fact, it is also intractable to compute mean vectors and covariance matrixes in feature space because feature space has a quite high or even infinite

dimension. Fortunately, with kernel trick embedded, computing on the kernel matrix is quite feasible since the dimension of the principal subspace is bounded by the data size N.

Consequently, the parameters that kernel GMM needs to estimate are the prior probability $\alpha_l^{(t)}$, centered kernel matrix $\tilde{\mathbf{K}}_l^{(t)}$, centered projecting kernel matrix $(\tilde{\mathbf{K}}_l^{(t)})'$ and $w_l^{(t)}$ (see Table 1). That is to say, the M-components kernel GMM is determined by parameters $\theta_l = \left(\alpha_l^{(t)}, w_l^{(t)}, \tilde{\mathbf{K}}_l^{(t)}, (\tilde{\mathbf{K}}_l^{(t)})'\right)$, $l = 1, \cdots, M$.

According to the properties in previous sections, the EM algorithm for parameter estimation of kGMM could be summarized as in Table 2. Assuming the number of Gaussian components is M, we initialize the posterior probability p_{li} that each sample belongs to some Gaussian component. The algorithm could not be terminated until it converges or the presetting maximum iteration step is reached.

Table 2. Parameter Estimation Algorithm for kGMM

Step 0. Initialize all $p_{li}^{(0)}$ $(l = 1, \cdots, M;\ i = 1, \cdots, N)$, $t = 0$, set t_{max} and stopping condition false.

Step 1. While stopping condition is false, $t = t + 1$, do Step 2-7.

Step 2. Compute $\alpha_l^{(t)}$, $w_{li}^{(t)}$, $\mathbf{W}_l^{(t)}$, $(\mathbf{W}_l^{(t)})'$, $\mathbf{K}_l^{(t)}$ and $(\mathbf{K}_l^{(t)})'$ according to notations in Table 1.

Step 3. Compute the matrixes $\tilde{\mathbf{K}}_l^{(t)}$, $(\tilde{\mathbf{K}}_l^{(t)})'$ via Property 1.

Step 4. Compute the largest d_ϕ eigenvalues and eigenvectors of centered kernel matrixes $\tilde{\mathbf{K}}_l^{(t)}$.

Step 5. Compute $\hat{G}_l(\phi(x_j)|\theta_l^{(t)})$ via Property 3.

Step 6. Compute all posterior probabilities $p_{li}^{(t)}$ via (9)

Step 7. Test the stopping condition.

If $t > t_{\max}$ or $\sum_{l=1}^{k}\sum_{i=1}^{N}\left(p_{li}^{(t)} - p_{li}^{(t-1)}\right)^2 < \varepsilon$, set stopping condition true, otherwise loop back to *Step 1.*

3.4 Discussion of the Algorithm

Computational Cost and Speedup Techniques on Large Scale Problem. By employing kernel trick, the computational cost of kernel eigen-decomposition based methods is almost involved by the eigen-decomposition step. Therefore, the computational cost mainly depends on the size of kernel matrix, i.e. the size of data set. If the size N is not very large (e.g. $N \leq 1,000$), it is not a problem to obtain full eigen-decomposition. If the size N is large enough, it is liable to meet with the curse of dimension. As is known, if $N > 5,000$, it is impossible to finish full eigen-decomposition even within hours on the fastest PCs currently. However, the size N is usually very large in some problems such as data mining.

Fortunately, as we have pointed out in Section 3.2, we need not obtain the full eigen-decomposition for components of kGMM, and we only need estimate

the largest d_ϕ nonzero eigenvalues and corresponding eigenvectors. As for large scale problem, we could make the assumption that $d_\phi \ll N$. Some techniques can be adopted to estimate the largest d_ϕ components for kernel methods. The first technique is based on traditional Orthogonal Iteration or Lanzcos Iteration. The second is to make the kernel matrix sparse by sampling techniques [1]. The third is to apply Nyström method to speedup kernel machine [19].

However, these three techniques are a little complicated. In this paper, we adopt another much simple but practical technique proposed by Taylor et.al [20]. It assumes that all samples forming kernel matrix of each component are drawn from an underlying density $p(x)$. And the problem could be written down as a continue eigen-problem.

$$\int k(x,y)p(x)\beta_i(x)dx = \lambda_i\beta_i(y) \tag{17}$$

where λ_i, $\beta_i(y)$ are eigenvalue and eigenvector, and k is a given kernel function.

The integral could be approximate using Monte Carlo method by a subset of samples $\{x_i\}_{i=1}^m$ $(m \ll N, m \geqslant d_\phi)$ drawn according to $p(x)$.

$$\int k(x,y)p(x)V_i(x)dx \approx \frac{1}{N}\sum\nolimits_{j=1}^{N} k(x_j,y)V_i(x_j) \tag{18}$$

Plugging in $y = x_k$ for $j = 1, \cdots, N$, we obtain a matrix eigen-problem.

$$\frac{1}{N}\sum\nolimits_{j=1}^{N} k(x_j,x_k)V_i(x_j) = \hat{\lambda}_i V_i(x_k) \tag{19}$$

where $\hat{\lambda}_i$ is the approximation of eigenvalue λ_i.

This approximation approach has been proved feasible and has bounded error. We apply it to our parameter estimation algorithm on large scale problem $(N > 1,000)$. In our algorithm, the underlying density of component l is approximated by $\hat{G}_l(\phi(x)|\theta_l^{(t)})$. We do sampling to obtain a subset with size m according to $\hat{G}_l(\phi(x)|\theta_l^{(t)})$, and perform full eigen-decomposition on such subset to obtain the largest d_ϕ eigen-components.

With employing this Monte Carlo sampling technique, the computational cost upon large scale problem could be reduced greatly. Furthermore, the memory needed by the parameter estimation algorithm also reduces greatly upon large scale problem. These makes the proposed kGMM efficient and practical.

Comparison with Related Works. There are still some other work related to ours. One major is the spectral clustering algorithm [21]. Spectral clustering could be regarded as using RBF based kernel method to extract features and then performing clustering by K-means. Compared with spectral clustering, the proposed kGMM has at least two advantages. (1)kGMM can provide result in probabilistic framework and can incorporate prior information easily. (2) kGMM can be used in supervised learning problem as a density estimation method. All these advantages encourage us to apply the proposed kGMM.

Misunderstanding. We emphasize a misunderstanding of the proposed model. Someone doubt that it can simply run GMM in the reduced dimension space obtained by Kernel PCA to achieve the same result as kGMM. They say that this just need project the data into the first d_ϕ dimension of the feature space by Kernel PCA, and then perform GMM parameter estimation in that principle subspace. However, the choice of a proper d_ϕ is a critical problem so that the performance will completely depend on the choice of d_ϕ. If d_ϕ is too large, the estimation is not feasible because probability estimation demands that the number of samples is large enough in comparison with dimension d_ϕ, and the computational cost will increase greatly simultaneously. On the contrary, small d_ϕ makes the estimated parameters not "well represent" the data.

The proposed kGMM does not have that problem since the approximated density function is complete and optimal under the minimum Kullback-Leibler divergence criteria. Moreover, kGMM can allow different component with different d_ϕ. All these improve the flexibility and expand the application of the proposed kGMM.

4 Experiments

In this section, two experiments are performed to validate the proposed kGMM compared with traditional GMM. Firstly kGMM is employed as unsupervised learning or clustering method on synthetic data set. Secondly kGMM is employed as supervised density estimation method for real-world handwritten digit recognition.

4.1 Synthetic Data Clustering Using Kernel GMM

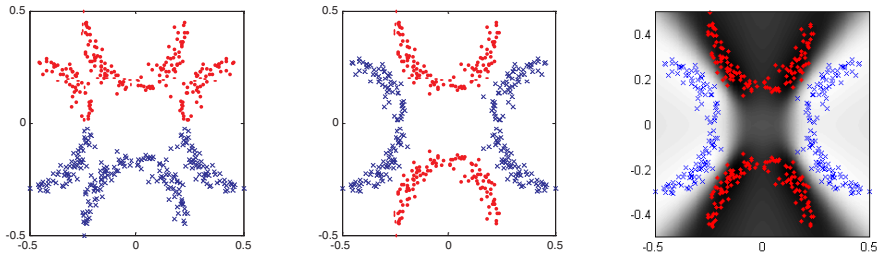
To provide an intuitive comparison between the proposed kGMM and traditional GMM, we first conduct experiments on synthetic 2-D data sets.

The data sets each with 1,000 samples are depicted in Figure 1 and Figure 2. For traditional GMM, all the samples are used to estimate the parameters of two components mixture of Gaussian. When the algorithm stops, each sample will belong to one of the components or clusters according to its posterior. The clustering results of traditional GMM are shown in Figure 1(a) and Figure 2(a). The results are obviously not satisfying.

However, by using kGMM with a polynomial kernel of degree 2, $d_\phi = 4$ for each Gaussian component and the same clustering scheme as traditional GMM, we achieve the promising results as shown in Figure 1(b) and Figure 2(b). Besides, kGMM provides probabilistic information as in Figure 1(c) and Figure 2(c), which cannot provide by most classical kernel methods.

4.2 USPS Data-Set Recognition Using Kernel GMM

Kernel GMM is also applied to a real-world problem, the US Postal Service (USPS) handwritten digit recognition. The data set consists of 9,226 grayscale



(a) (b) (c)

Fig. 2. Data set consists of 1,000 samples. Points marked by '×' belong to one cluster and marked by '·' belong to the other. (a) is the partition result by traditional GMM; (b) is the result achieved by kGMM; (c) shows the probability that each point belongs to the left-right cluster. The whiter the point is, the higher the probability is.

images of size 16x16, divided into a training set of 7,219 images and a test set of 2,007 images.

The original input data is just vector form of the digit image, i.e., the input feature space is with dimensionality 256. Optionally, we can perform a linear discriminant analysis (LDA) to reduce the dimensionality of feature space. If LDA is performed, the feature space yields to be 39.

Each category ω is estimated a density of $p(x|\omega)$ using 4 components GMM on training set. To classify an test sample x, we use the Bayesian decision rule

$$\omega* = \arg\max_{\omega} \left\{ p(x|\omega)P(\omega) \right\}, \ \omega = 1, \cdots, 10 \tag{20}$$

where $P(\omega)$ is prior probability of category ω. In this experiment, we set $P(\omega) = 1/10$. That is to say, all categories are with equal prior probability.

To be comparison, kGMM adopts the same experiment scheme as traditional GMM except for using an RBF kernel function

$$k(x, x') = \exp(-\gamma \|x - x'\|^2)$$

with $\gamma = 0.0015$, Gaussian mixture component number of 2 and $d_\phi = 40$ for each Gaussian component.

The experiment results of GMM in original input space, GMM in the space by LDA (from [4]) and kGMM are shown in Table 3. We can see that kGMM, with less components number, has obviously much better performance than traditional GMM with or without LDA. Although, the result by kGMM is not the state-of-art result on USPS, we still can improve the result by incorporating invariance prior knowledge using Tangent distance as in [4].

Table 3. Comparison results on USPS data set

Method	Best Error rate
GMM	8.0%
LDA+GMM	6.7%
Kernel GMM	4.3%

5 Conclusion

In this paper, we present a kernel Gaussian Mixture Model, and deduce a parameter estimation algorithm by embedding kernel trick into EM algorithm. Furthermore, we adopt a Monte Carlo sampling technique to speedup kGMM upon large scale problem, thus make it more practical and efficient.

Compared with most classical kernel methods, kGMM can solve problems in a probabilistic framework. Moreover, it can tackle nonlinear problems better than the traditional GMM. Experimental results on synthetic and real-world data set show that the proposed approach has satisfied performance.

Our future work will focus on incorporating prior knowledge such as invariance in kGMM and enriching its applications.

Acknowledgements. The author would like to thank anonymous reviewers for their helpful comments, also thank Jason Xu for helpful conversations about this work.

References

1. Achlioptas, D., McSherry, F. and Schölkopf, B.: Sampling techniques for kernel methods. In Advances in Neural Information Processing System (NIPS) 14, MIT Press, Cambridge MA (2002)
2. Bilmes, J. A.: A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, Technical Report, UC Berkeley, ICSI-TR-97-021 (1997)
3. Bishop, C. M.: Neural Networks for Pattern Recognition, Oxford University Press. (1995)
4. Dahmen, J., Keysers, D., Ney, H. and Güld, M.O.: Statistical Image Object Recognition using Mixture Densities. Journal of Mathematical Imaging and Vision, 14(3) (2001) 285–296
5. Duda, R. O., Hart, P. E. and Stork, D. G.: Pattern Classification, New York: John Wiley & Sons Press, 2nd Edition. (2001)
6. Everitt, B. S.: An Introduction to Latent Variable Models, London: Chapman and Hall. (1984)
7. Francis R. B. and Michael I. J.: Kernel Independent Component Analysis, Journal of Machine Learning Research, 3, (2002) 1–48
8. Gestel, T. V., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., Moor, B. De and Vanderwalle J.: Bayesian framework for least squares support vector machine classifiers, gaussian processs and kernel fisher discriminant analysis. Neural Computation, 15(5) (2002) 1115–1148

9. Herbrich, R., Graepel, T. and Campbell, C.: Bayes Point Machines: Estimating the Bayes Point in Kernel Space. In Proceedings of International Joint Conference on Artificial Intelligence Work-shop on Support Vector Machines, (1999) 23–27
10. Kwok, J. T.: The Evidence Framework Applied to Support Vector Machines, IEEE Trans. on NN, Vol. 11 (2000) 1162–1173.
11. Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.R.: Fisher discriminant analysis with kernels. IEEE Workshop on Neural Networks for Signal Processing IX, (1999) 41–48
12. Mjolsness, E. and Decoste, D.: Machine Learning for Science: State of the Art and Future Pros-pects, Science. Vol. 293 (2001)
13. Roberts, S. J.: Parametric and Non-Parametric Unsupervised Cluster Analysis, Pattern Recogni-tion, Vol. 30. No 2, (1997) 261–272
14. Schölkopf, B., Smola, A.J. and Müller, K.R.: Nonlinear Component Analysis as a Kernel Eigen-value Problem, Neural Computation, 10(5), (1998) 1299–1319
15. Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K. R., Raetsch, G. and Smola, A.: Input Space vs. Feature Space in Kernel-Based Methods, IEEE Trans. on NN, Vol 10. No. 5, (1999) 1000–1017
16. Schölkopf, B. and Smola, A. J.: Learning with Kernels: Support Vector Machines, Regularization and Beyond, MIT Press, Cambridge MA (2002)
17. Tipping, M. E.: Sparse Bayesian Learning and the Relevance Vector Machine, Journal of Machine Learning Research. (2001)
18. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd Edition, Springer-Verlag, New York (1997)
19. Williams, C. and Seeger, M.: Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Diettrich, and V. Tresp, editors, Advances in Neural Information Processing Systems (NIPS)13. MIT Press, Cambridge MA (2001)
20. Taylor, J. S., Williams, C., Cristianini, N. and Kandola J.: On the Eigenspectrum of the Gram Matrix and Its Relationship to the Operator Eigenspectrum, N. CesaBianchi et al. (Eds.): ALT 2002, LNAI 2533, Springer-Verlag, Berlin Heidelberg (2002) 23–40
21. Ng, A. Y., Jordan, M. I. and Weiss, Y.: On Spectral Clustering: Analysis and an algorithm, Advance in Neural Information Processing Systems (NIPS) 14, MIT Press, Cambridge MA (2002)
22. Moghaddam, B. and Pentland, A.: Probabilistic visual learning for object representation, IEEE Trans. on PAMI, Vol. 19, No. 7 (1997) 696–710

Appendix

To be convenient, the subscripts representing Gaussian components and the superscripts representing iterations are omitted in this part.

[Lemma 1] Suppose $\phi(\cdot)$ is a mapping function that satisfies Mercer conditions as in Equation (1)(Section 2) with N training samples $X = \{x_i\}_{i=1}^N$. w is a N-dimensional column vector with $w = [w_1, \cdots, w_N]^T \in \mathbf{R}^N$.

Let's define $\mu = \sum_{i=1}^N \phi(x_i) w_i^2$ and $\tilde{\phi}(x_i) = \phi(x_i) - \mu$

Then the following consequences hold true:

(1) If $\mathbf{K}$ is a $N \times N$ kernel matrix such that $\mathbf{K}_{ij} = \big(w_i \phi(x_i) \cdot w_j \phi(x_j)\big)$, and $\tilde{\mathbf{K}}$ is a $N \times N$ matrix, which is centered in the feature space, such that

$\tilde{\mathbf{K}}_{ij} = \big(w_i\tilde{\phi}(x_i) \cdot w_j\tilde{\phi}(x_j)\big)$, then we can get:

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{WK} - \mathbf{KW} + \mathbf{WKW} \tag{a1}$$

where $\mathbf{W} = ww^T$.

(2) If $\mathbf{K}'$ is a $N \times N$ projecting kernel matrix such that $\mathbf{K}'_{ij} = \big(\phi(x_i) \cdot w_j\phi(x_j)\big)$, and $\tilde{\mathbf{K}}'$ is a $N \times N$ matrix, which is centered in the feature space, such that $\widetilde{\mathbf{K}}'_{ij} = \big(\tilde{\phi}(x_i) \cdot w_j\tilde{\phi}(x_j)\big)$, then

$$\tilde{\mathbf{K}}' = \mathbf{K}' - \mathbf{W}'\mathbf{K} - \mathbf{K}'\mathbf{W} + \mathbf{W}'\mathbf{KW} \tag{a2}$$

where $\mathbf{W}' = \mathbf{1}_N w^T$, $\mathbf{1}_N$ is a N-dimensional column vector that all entries equal to 1.

Proof: (1)

$$\begin{aligned}
\widetilde{\mathbf{K}}_{ij} &= \big(w_i\tilde{\phi}(x_i) \cdot w_j\tilde{\phi}(x_j)\big) \\
&= w_i\tilde{\phi}(x_i)^T w_j\tilde{\phi}(x_j) \\
&= \big(w_i(\phi(x_i) - \sum_{k=1}^{N}\phi(x_k)w_k^2)\big)^T\big(w_j(\phi(x_j) - \sum_{k=1}^{N}\phi(x_k)w_k^2)\big) \\
&= \big(w_i\phi(x_i)^T w_j\phi(x_j)\big) - w_i\sum_{k=1}^{N} w_k\big(w_k\phi(x_k)^T w_j\phi(x_j)\big) \\
&\quad - \sum_{k=1}^{N} w_k\big(w_i\phi(x_i)^T w_k\phi(x_k)\big)\cdot w_j + w_i\sum_{k=1}^{N}\sum_{n=1}^{N} w_k w_n\big(w_k\phi(x_k)^T w_n\phi(x_n)\big) \\
&= \mathbf{K}_{ij} - w_i\sum_{k=1}^{N} w_k\mathbf{K}_{kj} - \sum_{k=1}^{N} w_k\mathbf{K}_{ik}w_j + w_i\sum_{k=1}^{N}\sum_{n=1}^{N} w_k w_n\big(w_k\phi(x_k)^T w_n\phi(x_n)\big)w_j
\end{aligned}$$

then we can get the more compact expression as (a1).

Similarly, we can prove (2).

[Lemma 2] Suppose Σ is a covariance matrix such that

$$\Sigma = \sum\nolimits_{i=1}^{N} \tilde{\phi}(x_i)\tilde{\phi}(x_i)^T w_i^2 \tag{a3}$$

Then the following would be hold

(1) $\Sigma V = \lambda V \Leftrightarrow \tilde{\mathbf{K}}\beta = \lambda\beta$.

(2) $V = \sum_{i=1}^{\mathrm{N}} \beta_i\tilde{\phi}(x_i)w_i$.

Proof:

(1) Firstly, we prove " $\Rightarrow$ ".

If $\Sigma V = \lambda V$, then the solution V lies in space spanned by $w_1\tilde{\phi}(x_1), \cdots, w_N\tilde{\phi}(x_N)$, and we have two useful consequences: firstly, we can consider the equivalent equation

$$\lambda\big(w_k\tilde{\phi}(x_k) \cdot V\big) = \big(w_k\tilde{\phi}(x_k) \cdot \Sigma V\big), \text{ for all } k = 1, \cdots, N \tag{a4}$$

and secondly, there exist coefficients $\beta_i \quad (i = 1, \cdots, N)$ such that

$$V = \sum\nolimits_{i=1}^{N} \beta_i w_i \tilde{\phi}(x_i) \tag{a5}$$

Combining (a4) and (a5), we get

$$\lambda \sum_{i=1}^{N} \beta_i \big(w_k \tilde{\phi}(x_k) \cdot w_i \tilde{\phi}(x_i)\big) = \sum_{i=1}^{N} \beta_i \Big(w_k \tilde{\phi}(x_k) \cdot \sum_{j=1}^{N} w_j \tilde{\phi}(x_j) \big(w_j \tilde{\phi}(x_j) \cdot w_i \tilde{\phi}(x_i)\big)\Big)$$

then this can read

$$\lambda \tilde{K} \beta = \tilde{K}^2 \beta$$

where $\beta = [\beta_1, \cdots, \beta_N]^T$. $\tilde{K}$ is a symmetric matrix, which has a set of eigenvectors which span the whole space, thus

$$\lambda \beta = \tilde{K} \beta \tag{a6}$$

Similarly, we can prove " $\Leftarrow$ ".
(2) is easy to prove, so the proof is omitted.

[Lemma 3] If $x \in \mathbf{R}^d$ is a sample, with $\tilde{\phi}(x) = \phi(x) - \mu$, then

$$\big(V \cdot \tilde{\phi}(x_j)\big) = \sum\nolimits_{i=1}^{N} \beta_i \big(w_i \tilde{\phi}(x_i) \cdot \tilde{\phi}(x_j)\big) = \beta^T \Gamma \tag{a7}$$

where Γ is *j-th* column of centered projecting matrix $\tilde{\mathbf{K}}'$.
Proof:
This Lemma follows from (a5).