A Kernel Between Unordered Sets of Data: The Gaussian Mixture Approach

Siwei Lyu

Department of Computer Science,
Dartmouth College, Hanover, NH 03755, USA

Abstract. In this paper, we present a new kernel for unordered sets of data of the same type. It works by first fitting a set with a Gaussian mixture, then evaluate an efficient kernel on the two fitted Gaussian mixtures. Furthermore, we show that this kernel can be extended to sets embedded in a feature space implicitly defined by another kernel, where Gaussian mixtures are fitted with the kernelized EM algorithm [6], and the kernel for Gaussian mixtures are modified to use the outputs from the kernelized EM. All computation depends on data only through their inner products as evaluations of the base kernel. The kernel is computable in closed form, and being able to work in a feature space improves its flexibility and applicability. Its performance is evaluated in experiments on both synthesized and real data.

1 Introduction

Kernel methods received attention originally as a "trick" to introduce non-linearity into the support vector machines (SVM) [20]. Evaluating a kernel function between two data is equivalent to computing the inner product of their images in a non-linearly mapped Hilbert space (the feature space). It is realized later that kernel methods are more general: similar to SVMs, many other linear algorithms also depend on data through their inner products. By substituting the inner products with kernel evaluations, these linear algorithms assume power to discover non-linear patterns in data [18]. Recent years have seen significant development in "kernelizing" existing algorithms, examples include kernel PCA [15], kernel FLD [14] and kernel k-means [1]. The kernelized algorithms inherit the innate stability of their linear ancestors, thus largely reduce the possibility of over-fitting the training data.

One important advantage of the kernel methods [18] is that they enable algorithms originally designed for vectors of finite dimensions (e.g., PCA, FLD or SVM) to work with discrete, structured or infinite dimensional data types, such as strings [21], statistical manifolds [7] and graphs [11]. With properly designed kernels, these data types are implicitly embedded into a vector space and lend themselves to kernel-based algorithms.

In this paper, we present a kernel for unordered sets of data of the same type, which are useful data models in many applications. For instance, in document categorization, documents are usually represented as "bag-of-words", which are

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 255–267, 2005.

unordered set of key words. Images can also be treated as "bag-of-tuples", where the element is the tuple of the position and intensity of a pixel in an image[8]. Instead of directly defining a kernel between two sets, we take the methodology of first modeling each set probabilistically, and then constructing a kernel between the two probabilistic models. More specifically, each set is treated as a collection of i.i.d. samples from an unknown probability distribution, whose probability density function (pdf) is taken from a parametric family. The kernel between two sets is thus computed as evaluating a kernel between the two pdfs. In this paper, we employ Gaussian mixtures to model the generating pdf of a vector set. On the two estimated Gaussian mixtures, the (normalized) expected likelihood kernel is evaluated, which affords an efficient computation without integration. Furthermore, the Gaussian mixture fitting and kernel evaluation are extended to a feature space implicitly defined by another kernel. The proposed method is evaluated on both synthesized and real data sets.

2 Kernel Function and Kernel-Induced Feature Space

Given an input space $\mathcal{X}$, a kernel K is a function $K(x,z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{H}}$ for any $x, z \in \mathcal{X}$, where ϕ is a mapping from $\mathcal{X}$ to a Hilbert space $\mathcal{H}$ (the feature space), and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product operator in $\mathcal{H}$ [18]. Admissible kernel can be specified without implicit reference to $\mathcal{H}$ or ϕ with the finite positive definite property: any real-valued symmetric binary function on $\mathcal{X}$ is a kernel if it satisfies the finite positive definite property:

$$\sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) \geq 0$$

for any $m \in \mathbb{N}$, any subset $\{x_1, \cdots, x_m\}$ of $\mathcal{X}$ and any choice of real numbers $c_1, \cdots, c_m$. Equivalently, the finite positive definite property can be expressed as that the matrix formed by restricting the kernel function on any finite subset of $\mathcal{X}$ is positive semi-definite. There is an equivalence between a kernel function $K(\cdot, \cdot)$ and a corresponding kernel-induced feature space $\mathcal{H}$: any admissible kernel function also ensures the existence of a feature space and vice versa.

3 The Gaussian Mixture Model

We consider unordered sets of d-dimensional vectors, $\chi = \{x_1, \cdots, x_N\}$. One can model the data in χ as *i.i.d* samples from a multivariate Gaussian distribution $G(x; \mu, C) = \frac{1}{(2\pi)^{\frac{d}{2}} |C|^{\frac{1}{2}}} \exp(-\frac{1}{2}(x-\mu)^T C^{-1} (x-\mu))$, parameterized by the mean μ and the covariance matrix, C. Parameters μ and C are estimated from data with the sample mean $\bar{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$ and the empirical covariance matrix $\bar{C} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{\mu})(x_i - \bar{\mu})^T$, respectively. Major advantages of Gaussian is simplicity. However, a Gaussian cannot model a multi-modal distribution, which usually is the case in practice. In this aspect, a Gaussian mixture has much more modeling flexibility. A finite Gaussian mixture is defined as:

$$p(x) = \sum_{k=1}^{M} \alpha_k G(x; \mu_k, C_k), \tag{1}$$

where $M \in \mathbb{N}$ is the number of components, $\alpha_1, \cdots, \alpha_M$ are the mixing coefficients satisfying $\sum_{k=1}^{M} \alpha_k = 1$ and $\alpha_i \geq 0$ for $i = 1, \cdots M$. Parameters μ_k and C_k are the mean and covariance of each Gaussian in the mixture. It can be shown that [16] with a sufficient number of components, any probability density can be approximated to any degree by a Gaussian mixture.

The parameters in a Gaussian mixture, the mixing coefficients, $\alpha_1, \cdots, \alpha_M$, the mean and covariance of each component, $\mu_1, \cdots, \mu_M$ and $C_1, \cdots, C_M$, can be estimated from set χ with the expectation-maximization (EM) algorithm [2], given that the number of components M is known. Starting from initial values of these parameters, the EM algorithm proceeds by executing the following steps until convergence,

$$p_k(i) = \frac{\alpha_k G(x_i; \mu_k, C_k)}{\sum_{j=1}^{M} \alpha_j G(x_i; \mu_j, C_j)} \quad \text{for } i = 1, \cdots, N, k = 1, \cdots, M \tag{2}$$

$$\alpha_k = \tfrac{1}{N} \textstyle\sum_{i=1}^{N} p_k(i), \text{ for } k = 1, \cdots, M \tag{3}$$

$$\mu_k = \frac{\sum_{i=1}^{N} x_i p_k(i)}{\sum_{i=1}^{N} p_k(i)}, \text{ for } k = 1, \cdots, M \tag{4}$$

$$C_k = \frac{\sum_{i=1}^{N} (x_i - \mu_k)(x_i - \mu_k)^T p_k(i)}{\sum_{i=1}^{N} p_k(i)}, \text{ for } k = 1, \cdots, M. \tag{5}$$

The EM algorithm guarantees to converge within finite steps to a local maximum of the log-likelihood function of the parameters given data χ. More details of the EM estimation for Gaussian mixtures can be found in [2].

4 Kernels Between Sets of Vectors

A kernel function for unordered sets of d-dimensional vectors, $\chi = \{x_1, \cdots, x_N\}$, can be built from the probabilistic modeling of the set. Specifically, each set can be treated as a collection of i.i.d. samples from a probability distribution, whose density function (pdf) is approximated with a parametric family $\mathcal{P}$. A kernel between the estimated two pdfs can be defined and used as the kernel between the two sets. With the estimated pdfs, generally, any similarity measures between two pdfs, such as the Jensen-Shannon divergence, Kullback-Leibler divergence or the χ^2 distance, can be used to construct kernels between two pdfs [5]. The problem is that such measures may not be efficiently computable, especially in high-dimensional data spaces. In a related work [10], $\mathcal{P}$ was chosen as the multivariate Gaussian distributions. Then the Bhattacharyya kernel,

$K_B(p,q) = \int_{\mathcal{X}} \sqrt{p(x)}\sqrt{q(x)}dx$, was computed between the two Gaussian distributions. This kernel is a special case of the more general class of probability product kernels [9], which is defined as $K_{PP}(p,q) = \int_{\mathcal{X}} p(x)^\rho q(x)^\rho dx$, with $\rho = 1/2$. For pdfs in the exponential family (multivariate Gaussian as a special case), the probability product kernels can be computed efficiently without integration. However, when the two distributions are Gaussian mixtures, the general probability product kernels do not give rise to efficient evaluation, as numerical integration can not be avoided.

5 Expected Likelihood Kernel Between Gaussian Mixtures

In this work, we employ the expected likelihood kernel between the two estimated Gaussian mixtures, as the results of running the EM algorithm on the two unordered sets of d-dimensional vectors. The expected likelihood kernel is defined as:

$$K_{EL}(p,q) = \int_{\mathcal{X}} p(x)q(x)dx, \tag{6}$$

which is seen to be a special case of the probability product kernel with $\rho = 1$. As formally stated in the following theorem, the expected likelihood kernel affords an efficient computation for Gaussian mixtures.

Theorem 1. *For two Gaussian mixtures of d dimensional real random vectors,*

$$p(x) = \textstyle\sum_{k=1}^{M_1} \alpha_k^{(1)} G(x; \mu_k^{(1)}, C_k^{(1)}) \qquad \text{and} \qquad q(x) = \textstyle\sum_{k=1}^{M_2} \alpha_k^{(2)} G(x; \mu_k^{(2)}, C_k^{(2)}),$$

the expected likelihood kernel, Eq.(6), is computed as

$$K_{EL}(p,q) = (2\pi)^{-\frac{d}{2}} \alpha^T \Gamma \beta,$$

for $\alpha = (\alpha_1^{(1)}, \cdots, \alpha_{M_1}^{(1)})^T$ and $\beta = (\alpha_1^{(2)}, \cdots, \alpha_{M_2}^{(2)})^T$. The $M_1 \times M_2$ matrix Γ is formed as $(\Gamma)_{ij} = g(\mu_i^{(1)}, C_i^{(1)}, \mu_j^{(2)}, C_j^{(2)})$, where function g is defined as:

$$g\left(\mu_1, C_1, \mu_2, C_2\right) = \frac{|C|^{\frac{1}{2}} \exp(\frac{1}{2}\mu^T C \mu)}{\prod_{i=1}^2 |C_i|^{\frac{1}{2}} \exp(\frac{1}{2}{\mu_i}^T {C_i}^{-1} \mu_i)}, \tag{7}$$

with $\mu = {C_1}^{-1}\mu_1 + {C_2}^{-1}\mu_2$ and $C = \left({C_1}^{-1} + {C_2}^{-1}\right)^{-1}$.

Proof. First, the integration of the product of two Gaussians, $G(x;\mu_1,C_1)$ and $G(x;\mu_2,C_2)$, is computed as [17]:

$$\int_{\mathbb{R}^d} G(x;\mu_1,C_1) \times G(x;\mu_2,C_2)dx = (2\pi)^{-\frac{d}{2}} g\left(\mu_1, C_1, \mu_2, C_2\right). \tag{8}$$

Substituting Eq.(8) into Eq.(6) and interchange the order of addition and multiplication (using Fubini's theorem) yields

$$\int_{\mathbb{R}^d} p(x)q(x)dx = \int_{\mathbb{R}^d} \sum_{i=1}^{M_1} \alpha_i^{(1)} G(x; \mu_i^{(1)}, C_i^{(1)}) \times \sum_{j=1}^{M_2} \alpha_j^{(2)} G(x; \mu_j^{(2)}, C_j^{(2)}) dx$$

$$= \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \alpha_i^{(1)} \alpha_j^{(2)} \int_{\mathbb{R}^d} G(x; \mu_i^{(1)}, C_i^{(1)}) \times G(x; \mu_j^{(2)}, C_j^{(2)}) dx$$

$$= \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \alpha_i^{(1)} \alpha_j^{(2)} (2\pi)^{-\frac{d}{2}} g\left(\mu_i^{(1)}, C_i^{(1)}, \mu_j^{(2)}, C_j^{(2)}\right) = (2\pi)^{-\frac{d}{2}} \alpha^T \Gamma \beta. \qquad \square$$

Theorem 1 shows that one can evaluate the expected likelihood kernel on two Gaussian mixtures in closed-form without integration. The kernel can further be made independent of the dimensionality of data, if we use its normalization: $K_{NEL}(p,q) = \frac{K_{EL}(p,q)}{\sqrt{K_{EL}(p,p)}\sqrt{K_{EL}(q,q)}}$, which can be shown to be an admissible kernel function. This property is essential when we extend to Gaussian mixtures estimated in a kernel-induced feature space, where the dimensionality of the data is usually not known.

6 Kernels Between Sets in Feature Space

The expected likelihood kernel evaluation for unordered sets can be extended to data in a feature space implicitly defined by another base kernel κ. This is the case when the sets contain non-vectorial data, the base kernel is used to implicitly map them into a vector space. For vectors, such implicit nonlinear mapping is also desirable when nonlinear data patterns are sought. Our basic methodology stays the same: Gaussian mixtures are first fitted to data sets and the (normalized) expected likelihood kernel is evaluated between the two fitted Gaussian mixtures. What differs is that all steps are implicitly performed in a feature space.

Working in a feature space poses two fundamental difficulties for the kernel evaluation described in Section 5. First, we may not fully recover a Gaussian in a feature space from a finite set. This is especially true when the dimension of the feature space is larger than the number of data in the set - only the partial covariance of each Gaussian restricted in the subspace spanned by the data (with a rank up to the size of the set) can be recovered. Another difficulty is that we usually do not have direct access to individual data except their inner products, computed with the evaluation of the base kernel. This renders the EM algorithm and the evaluation of the expected likelihood kernel not directly applicable: the estimation of the mean and covariance of each Gaussian (Eq.(4) and (5)) depend on individual data.

In face of these problems, the kernel evaluation is modified in the following aspects. First, in both the EM algorithm and the kernel evaluation, in lieu of the determinants and inverses of the full covariance matrices, the pseudo-determinants and pseudo-inverse [3] of the partial rank-deficient covariance matrix are used to avoid recovering the full covariance matrices. The pseudo-determinant of a matrix is the product of all its non-zero singular values[1] and its pseudo-inverse is obtained by inverting all nonzero singular values in its singular value decomposition. A property of the pseudo-determinant and pseudo-inverse important for the computation hereafter is Lemma 1. Due to limit of space, its is not presented here and can be found in the longer version of this paper [13].

Lemma 1. *If $C = XRR^TX^T$, and R and X^TX are invertible, then we have*

$$d(C) = |\tilde{R}^TX^TX\tilde{R}| \qquad \text{and} \qquad C^\dagger = X\tilde{R}\tilde{R}^TX^T,$$

where $\tilde{R} = (R^TX^TX)^{-1}$ and $\dagger$ is the pseudo-inverse operator.

Another change is that both the EM algorithm and the (normalized) expected likelihood kernel are reformulated to depends only on base kernel evaluation. Specifically, the Gaussian mixtures are fitted with a variant of the kernelized EM [6] and the evaluation of the (normalized) expected likelihood kernel is modified to use the outputs of the kernelized EM.

To avoid clumsiness in notation, hereafter in this section we still describe the reformulated algorithms in a vector space, bearing in mind that the inner products of these vectors will be replaced with evaluations of the base kernel in the feather space. Specifically, each datum is represented as a column vector, and a data set $\chi = \{x_1, \cdots, x_N\}$ is a matrix $X = [x_1, \cdots, x_N] \in \mathcal{R}^{d\times N}$. For two different data sets, X_1 and X_2, their inner product matrix $X_1^TX_2$ contains inner products between all pairs of data from the two sets.

6.1 Kernelized EM

To make the EM algorithm, Eq.(2)-(5), depend on inner products of data while being independent of individual data, we view it from an alternative perspective. Rewriting Eq.(4) and (5), the updating steps of the mean and covariance of each Gaussian in the mixture become

$$\mu_k = \textstyle\sum_{i=1}^N x_i w_k(i), \tag{9}$$
$$C_k = \textstyle\sum_{i=1}^N (x_i - \mu_k)(x_i - \mu_k)^T w_k(i), \tag{10}$$

for $k = 1, \cdots, M$, where $w_k(i) = \frac{p_k(i)}{\sum_{i=1}^N p_k(i)}$ is the weight associated with each Gaussian component and each datum in the set. Denote $w_k{=}[w_k(1),\cdot\cdot\cdot, w_k(N)]^T$. Eq.(9) and (10) can be rewritten more compactly as:

$$\mu_k = Xw_k, \tag{11}$$
$$C_k = X(I_N - w_k\mathbf{1}_N{}^T)\,\mathrm{diag}(w_k)(I_N - \mathbf{1}_Nw_k^T)X^T. \tag{12}$$

[1] More formally, $d(C) = \prod_{i=1}^n(\lambda_i + 1 - \mathrm{sign}(\lambda_i)^2)$ where λ_i is the singular value of matrix C and $\mathrm{sign}(x) = 1$ for positive x, and 0 for $x = 0$ and -1 for negative x.

where I_N is the $N \times N$ identity matrix and $\mathbf{1}_N$ is the N-dimensional column vector with all 1s. Operator diag$(\cdot)$ outputs a diagonal matrix whose diagonal is set to the input vector. Modulo to the data matrix X, the estimated mean and covariance of each Gaussian in the mixture are fully determined by weights w_k. EM can then be viewed as iteratively updating these weights in a bootstrapping fashion.

To update weights w_k, it is sufficient to compute $p_k(i)$ with Eq.(2). In a feature space, as suggested previously, the updating step is modified to use the pseudo-determinant and pseudo-inverse of the partial covariance matrices as:

$$p_k(i) = \frac{\alpha_k d(C_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^T C_k^{\dagger}(x_i - \mu_k)\right)}{\sum_{j=1}^{m} \alpha_j d(C_j)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_i - \mu_j)^T C_j^{\dagger}(x_i - \mu_j)\right)}. \tag{13}$$

The key step in updating $p_k(i)$ and hence w_k is to compute $d(C_k)$ and $(x_i - \mu_k)^T C_k^{\dagger}(x_i - \mu_k)$ for each data in the set and each Gaussian component. According to Lemma 1, the pseudo-determinant and pseudo-inverse of the partial covariance matrix C_k can be computed as:

$$d(C_k) = |R_k^T X^T X R_k| \tag{14}$$

$$C_k^{\dagger} = X R_k R_k^T X^T \tag{15}$$

where the $N \times N$ matrix R_k is

$$R_k = \left[(I_N - w_k \mathbf{1}_N{}^T)\sqrt{\mathrm{diag}(w_k)} X^T X\right]^{-1}, \tag{16}$$

with the square root computed component-wisely. Accordingly, $d(C_k)$ is computed from Eq.(14) and (16). With Eq.(15), term $(x_i - \mu_k)^T C_k^{\dagger}(x_i - \mu_k)$ is evaluated as

$$\begin{aligned}(x_i - \mu_k)^T C_k^{\dagger}(x_i - \mu_k) &= (x_i - X w_k)^T X R_k R_k^T X^T (x_i - X w_k) \\ &= \|R_k X^T X(\delta_i - w_k)\|^2,\end{aligned} \tag{17}$$

where δ_i is the N-dimensional column vector of all zeros except the i-th component being 1. $\|\cdot\|$ is the 2-norm of a vector. Note that no direct dependence on individual data or the data matrix appears in these computations. Matrix $X^T X$ is formed by inner products between each pair of data in X and is computed from evaluating the kernel matrix of κ on the input data set in the feature space.

In summarizing words, with initial values, the kernelized EM algorithm[2] proceeds by running the following steps until convergence

[2] The algorithm described here is in the same spirit as the original kernelized EM algorithm [6], differing in notations and the relaxed requirement of only recovering partial covariances.

- **step 1:** compute p_k with Eq.(17), (14) and (13);
- **step 2:** update weights w_k as $w_k(i) = \frac{p_k(i)}{\sum_{i=1}^{N} p_k(i)}$;
- **step 3:** update mixing coefficients α_k with Eq.(3);
- **step 4:** compute R_k with Eq.(16);

At the completion of the kernelized EM algorithm, the mixing coefficients α_k, vector w_k and matrix R_k[3] are output for each Gaussian in the mixture. It is from these outputs that the (normalized) expected likelihood kernel is evaluated in the feature space.

6.2 Kernel Evaluation

In the feature space, the expected likelihood kernel (and its normalization) is computed as in Theorem 1 with slight changes. First, the constant factor in the expected likelihood kernel is dropped, to yield $\widetilde{K_{EL}}(p,q) = \alpha^T \tilde{M} \beta$ on two Gaussian mixtures p and q. Denote X_1 and X_2 as the data matrices for the two mixtures. Vector α and β contains the mixing coefficients for p and q respectively. Matrix $\tilde{M}$ is formed as $(\tilde{M})_{ij} = \tilde{g}(\mu_i^{(1)}, C_i^{(1)}, \mu_j^{(2)}, C_j^{(2)})$ for each pair of Gaussians from the two mixtures. Function $\tilde{g}$ is defined as:

$$\tilde{g}\left(\mu_1, C_1, \mu_2, C_2\right) = \frac{d(C)^{\frac{1}{2}} \exp(\frac{1}{2}\mu^T C \mu)}{\prod_{i=1}^{2} d(C_k)^{\frac{1}{2}} \exp(\frac{1}{2}{\mu_k}^T {C_k}^{\dagger} \mu_k)}, \tag{18}$$

with $\mu = {C_1}^{\dagger}\mu_1 + {C_2}^{\dagger}\mu_2$ and $C = \left({C_1}^{\dagger} + C_2^{\dagger}\right)^{\dagger}$, which is computed in four steps.

Compute $d(C_k)$: From the outputs of the kernelized EM algorithm, $d(C_k)$ is computed with R_k and Eq.(14) as $d(C_k) = |R_k^T X^T X R_k|$, for $k = 1, 2$.

Compute ${\mu_k}^T {C_k}^{\dagger} \mu_k$: With R_k, Eq.(9) and (14), term ${\mu_k}^T {C_k}^{\dagger} \mu_k$ is computed as:

$${\mu_k}^T {C_k}^{\dagger} \mu_k = w_k^T X_k^T X_k R_k R_k^T X_k^T X_k w_k = \|R_k^T X_k^T X_k w_k\|^2 \tag{19}$$

for $k = 1, 2$.

Compute $d(C)$: With $C_k^{\dagger} = X_k R_k R_k^T X_k^T$ for $k = 1, 2$, we then have

$$C_1^{\dagger} + C_2^{\dagger} = X_1 R_1 R_1^T X_1^T + X_2 R_2 R_2^T X_2^T = [X_1 \ X_2] \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \begin{bmatrix} R_1^T & 0 \\ 0 & R_2^T \end{bmatrix} \begin{bmatrix} X_1^T \\ X_2^T \end{bmatrix}.$$

Now denote

$$R = \left(\begin{bmatrix} R_1^T & 0 \\ 0 & R_2^T \end{bmatrix} \begin{bmatrix} X_1^T X_1 & X_1^T X_2 \\ X_2^T X_1 & X_2^T X_2 \end{bmatrix} \right)^{-1} = \begin{bmatrix} R_1^T X_1^T X_1 & R_1^T X_1^T X_2 \\ R_2^T X_2^T X_1 & R_2^T X_2^T X_2 \end{bmatrix}^{-1} \tag{20}$$

and with Lemma 1, it holds that

[3] Note it is not necessary to output R_k as it can be computed from w_k with Eq.(16). However, it facilitates the evaluation of the kernel in next section.

$$C = \left(C_1{}^\dagger + C_2^\dagger\right)^\dagger = [X_1\ X_2]RR^T \begin{bmatrix} X_1^T \\ X_2^T \end{bmatrix}, \tag{21}$$

from which $d(C)$ is computed as

$$d(C) = \left| R^T \begin{bmatrix} X_1^T X_1 & X_1^T X_2 \\ X_2^T X_1 & X_2^T X_2 \end{bmatrix} R \right| \tag{22}$$

Compute $\mu^T C\mu$: Since we have $\mu = C_1^\dagger \mu_1 + C_2^\dagger \mu_2$, expanding $\mu^T C\mu$ yields

$$\mu^T C\mu = (C_1^\dagger\mu_1 + C_2^\dagger\mu_2)^T C (C_1^\dagger\mu_1 + C_2^\dagger\mu_2) = \sum_{i,j\in\{1,2\}} \mu_i^T {C_i^\dagger}^T C C_j^\dagger \mu_j. \tag{23}$$

Each term in the sum can be further expanded with $\mu_k = X_k w_k$ and $C_k^\dagger = X_k R_k R_k^T X_k^T$ for $k = 1, 2$, as:

$$\mu_i^T {C_i^\dagger}^T C C_j^\dagger \mu_j = w_i^T X_i^T X_i R_i R_i^T X_i^T [X_1\ X_2] RR^T \begin{bmatrix} X_1^T \\ X_2^T \end{bmatrix} X_j R_j R_j^T X_j^T X_j w_j$$

$$= w_i^T X_i^T X_i R_i R_i^T [X_i^T X_1\ \ X_i^T X_2] RR^T \begin{bmatrix} X_1^T X_j \\ X_2^T X_j \end{bmatrix} R_j R_j^T X_j^T X_j w_j. \tag{24}$$

All the above steps depend on data only through their inner products. Thus replacing these inner products with base kernel evaluations lead to computing function $\tilde{g}$ in the feature space with base kernel evaluations. Subsequently, the modified (normalized) expected likelihood kernel can also be computed based on kernel κ. Combining with the kernelized EM algorithm, this yields a kernel for unordered sets in the feature space.

7 Experiments

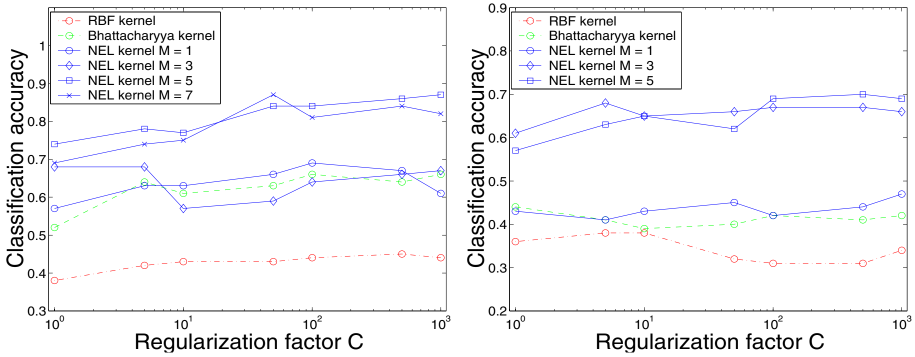
In this section, experimental results empirically evaluating the proposed kernel with other works are presented. The experiments were conducted on both synthesized and real data sets. In all experiments, the proposed kernel was coupled with SVM classifiers. Our SVM classifiers were implemented based on package LIBSVM [4] and were enhanced to work with kernels between sets. A simple multi-class classification protocol, a one-versus-the-rest scheme in training and a winner-takes-all strategy in testing was employed in classification.

7.1 Synthesized Data

Our synthesized data were $1,000$ sets of 5-D vectors of sizes ranging from 80 to 250 , each of which were random samples of one of the four different 5-D Gaussian mixtures with five components. All data sets were categorized into four different classes based on the Gaussian mixtures they were generated from. 700 out of the $1,000$ samples were used for training and the rest for testing. For a base of comparison, an SVM classifier with an RBF kernel $K_{RBF}(x, z) = \exp(-\frac{\|x-z\|^2}{\sigma^2})$ on 5-D vectors was trained. Each vector inherited class label of the set to which it belongs. The classification of a set was the result of a majority vote with the projected class labels of all its members. As no consideration is given to the correlation within a set, this plain RBF kernel did not perform well, as evident from Figure 7(a).

Another SVM classifier with a kernel between unordered sets as described in [10] was also compared, where a Gaussian was fitted to each set and the Bhattacharyya kernel $K_B(p,q) = \int_{\mathcal{X}} \sqrt{p(x)}\sqrt{q(x)}dx$ was evaluated on the two fitted Gaussians. For pdfs in the exponential family (Gaussian as a special case) it has been shown that the Bhattacharyya kernel afford a close-form evaluation [10]. However, for Gaussian mixtures with more than one components, the Bhattacharyya kernel loses that advantage, as the integration is not able to be removed. As shown in Figure 7(a), the Bhattacharyya kernel achieved a better performance than the simple RBF kernel. The improvement is most probably due to the better data modeling with a Gaussian fitting.

However, it is the normalized expected likelihood kernel with a proper number of components in the estimated Gaussian mixture (M = 5) that achieved the best overall performance, due to the more precise data modeling with Gaussian mixtures. For the other choices of component number M, with single component Gaussian mixture fitting, the normalized expected likelihood kernel is similar to the Bhattacharyya kernel (without the square root in the definition), which is reflected in their similar performance. For Gaussian mixtures with fewer components than the ground truth (as in the case of M = 3), the performance is not uniformly better than the base line case with M = 1. On the other hand, using more mixture components (e.g., M = 7) did not achieve significant improvement in performance yet the computational effort was increased. This bears the question of how to know the number of components in advance, as the EM can not be used to find it. Empirically, The number of components can be found by cross-validation. A more systematic approach is to use techniques that can automatically determine the number of components need on a data



(a) (b)

Fig. 1. Performance of the normalized expected likelihood kernel on the testing set for (a) synthesized data and (b) MNIST handwritten digit image dataset. Solid lines are for the Gaussian mixture with normalized expected likelihood kernel. Dashed lines are for the Bhattacharyya kernel.

set (e.g., [19]) and is left for future study. Also, in this experiment, for simplicity we fitted the same number of Gaussians to all sets. It is straightforward to extend to fitting Gaussian mixtures with different number of components to different sets.

7.2 Handwritten Digits Recognition

Our real data were a small subset of handwritten digit images. Specifically, we randomly chose 100 images for each of the 10 handwritten digit from the MNIST database [12], with 700 for training and the rest for testing. Similar to [10], each image was transformed to a set by sampling pixels with intensity greater than 191 on a 0 to 255 scale. The coordinates of these sampled pixel along with their intensities were presented to the algorithm. From each image, a set of size ranging from 50 to 108 with an average of 72 was obtained. Using a small subset of pixels is to avoid inverting a large inner product matrix, which is the most time consuming step in the kernel evaluation. Anticipating nonlinearity in data modeling, we chose an RBF kernel as the base kernel to nonlinearly map the 3-D tuples in each set into a feature space. To simplify the training process, we avoided extensive tuning of the parameter in the base kernel and in all cases the RBF kernel was set to have a width of $\sigma = 0.1$. The results were the average over 100 random splits of the training/testing splits of all the images chosen.

We trained again different SVM classifiers and their performances are shown in Figure 7(b). As in the case of the synthesized data, a simple RBF kernel led to the most inferior performance. However, the Bhattacharyya kernel with a RBF base kernel and the normalized expected likelihood kernel with a one-component Gaussian mixture modeling did not introduce much improvement, as a single Gaussian is not sufficient to model the generating probability distributions of these data sets. We then tested the normalized expected likelihood kernel with different number of mixture components (M = 3 and 5). Compared to other kernels, they achieved a substantial improvement in performance. Contrary to the previous case, it seems that the specific number of mixture components is somehow irrelevant in this case, as using 3 and 5 components did not result in significant difference in performance. We also observed that classification was relatively stable with regards to the regularization factor in the SVM classification.

8 Discussion

In this paper, we present a kernel between two unordered sets of data of the same type. Each set is first fitted with a Gaussian mixture. Then the expected likelihood kernel is evaluated between the two estimated Gaussian mixtures. Furthermore, this kernel function can be extended to cases when the data are in an implicitly defined feature space. The performance of this kernel is evaluated on both synthesized and real data sets. One drawback of the proposed algorithm, however, is running efficiency. Evaluating the kernel is quadratic in running time, which can be prohibitive in case of large data sets. We are working on approximation algorithms that achieve fast running time. Also, we are working on incorporating techniques that can automatically determine the number of components in a Gaussian mixture estimation.

References

1. A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. Journal of Machine Learning Research, 2(2):125-137, 2002.
2. J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, UC Berkeley, 1997.
3. S.L. Campbell and C.D. Meyer Jr. Generalized Inverses of Linear Transformations. Dover, New York, 1991.
4. C. Chang and C. Lin. LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
5. M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. Technical report, MPI for Biological Cybernetics, 2004.
6. J. Lee J. Wang and C. Zhang. Kernel trick embedded gaussian mixture model. In Lecture Notes in Artificial Intelligence, volume 2842, pages 159-174, 2003.
7. T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In Advances in Neural Information Processing Systems (NIPS), 1999.
8. T. Jebara. Image as bag of pixels. In International Conference on Computer Vision (ICCV), 2003.
9. T. Jebara, R. Kondor, and A. Howard. Probability product kernels. Journal of Machine Learning Research, 5, 2004.
10. R. Kondor and T. Jebara. A kernel between sets of vectors. In International Conference on Machine Learning (ICML), 2003.
11. R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In International Conference on Machine Learning (ICML), 2002.
12. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324, November 1998.
13. S. Lyu. Kernel between sets: the gaussian mixture approach. Technical Report TR2005-214, Computer Science Department, Dartmouth College, 2005.
14. S. Mika, G. Rütsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In IEEE Conference on Neural Networks for Signal Processing, 1999.
15. S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rütsch. Kernel PCA and de-noising in feature spaces. In Advances in Neural Information Processing Systems 11, pages 536-542, Cambridge, MA, 1999.
16. E. Parzen. On estimation of a probability density function and mode. Ann. Math. Statistics, 33:1,065-1,076, 1962.
17. S. Roweis. Gaussian Identities. Department of Computer Science, U. Toronto, Manuscript available at http://www.cs.toronto.edu/roweis/notes.html, 2001.
18. J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge, 2004.

19. C.A. Sugar and G.M. James. Finding the number of clusters in a dataset: An information-theoretic approach. Journal of American Statistical Association, 98(463):750-763, 2003.
20. V. Vapnik. Statistical Learning Theory. Wiley, New York, NY,1998.
21. S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In Advances in Neural Information Processing Systems (NIPS), 2003.