Tailoring Density Estimation via Reproducing Kernel Moment Matching

Le Song LE.SONG@NICTA.COM.AU
Xinhua Zhang XINHUA.ZHANG@NICTA.COM.AU
Alex Smola ALEX.SMOLA@NICTA.COM.AU
Statistical Machine Learning, NICTA, 7 London Circuit, Canberra, ACT 2601, Australia
Arthur Gretton ARTHUR.GRETTON@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tuebingen, Germany

Abstract

Moment matching is a popular means of parametric density estimation. We extend this technique to nonparametric estimation of mixture models. Our approach works by embedding distributions into a reproducing kernel Hilbert space, and performing moment matching in that space. This allows us to tailor density estimators to a function class of interest (i.e., for which we would like to compute expectations). We show our density estimation approach is useful in applications such as message compression in graphical models, and image classification and retrieval.

1. Introduction

Density estimation is a key element of statistician's toolbox, yet it remains a challenging problem. A popular class of methods relies on mixture models, such as Parzen windows (Parzen, 1962; Silverman, 1986) or mixtures of Gaussians or other basis functions (McLachlan & Basford, 1988). These models are normally learned using the likelihood. However, density estimation is often not the ultimate goal but rather an intermediate step in solving another problem. For instance, we may ultimately want to compute the expectation of a random variable or functions thereof. In this case it is not clear whether likelihood is the ideal objective, especially when the training sample size is small.

A second class of density estimators employ exponential family models and are based on the duality between maximum entropy and maximum likelihood estimation (Barndorff-Nielsen, 1978; Dudík et al., 2004; Altun & Smola, 2006). These methods match the moments of the estimators to those of the data, which helps focus the models on certain aspects of the data for particular applications. However, these parametric moment based methods can be too limited in terms of the class of distributions. Furthermore, exponential families tend to require highly nontrivial integration of high-dimensional distributions to ensure proper normalization. We desire to overcome these drawbacks and extend this technique to a larger class of models.

In this paper, we generalize moment matching to nonparametric mixture models. Our major aim is to tailor these density estimators for a particular function class, and provide uniform convergence guarantees for approximating the function expectations. The key idea is if we have good knowledge of the function class, we can tightly couple the density estimation with this knowledge. Rather than performing a full density estimation where we leave the function class and subsequent operations arbitrary, we restrict our attention to a smaller set of functions and the expectation operator. By exploiting this kind of domain knowledge, we make the hard density estimation problem easier.

Our approach is motivated by the fact that distributions can be represented as points in the marginal polytope in reproducing kernel Hilbert spaces (RKHS) (Wainwright & Jordan, 2003; Smola et al., 2007). By projecting data and density estimators into RKHS via kernel mean maps, we match them in that space (also referred to as the feature space). Choosing the kernel determines how much information about the density is retained by the kernel mean map, and thus which aspects (*e.g.*, moments) of a density are considered important in the matching process. The matching process, and thus our density estimation procedure, amounts to the solution of a convex quadratic program. We demonstrate the application of our approach in experiments, and show that it can lead to improvements in more complicated applications such as particle filtering and image processing.

2. Background

Let $\mathcal{X}$ be a compact domain and $X = \{x_1, \ldots, x_m\}$ be a sample of size m drawn independently and identically distributed (iid.) from a distribution p over $\mathcal{X}$. We aim to find an approximation $\hat{p}$ of p based on the sample X.

Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) on

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

$\mathcal{X}$ with kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and associated feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$. By design $\mathcal{H}$ has the reproducing property, that is, for any $f \in \mathcal{H}$ we have $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$. A kernel k is called *universal* if $\mathcal{H}$ is dense in the space of bounded continuous functions $C^0(\mathcal{X})$ on the compact domain $\mathcal{X}$ in the L_∞ norm (Steinwart, 2002). Examples of such kernels include Gaussian kernel $\exp(-\|x - x'\|^2 / 2\theta^2)$ and Laplace kernel $\exp(-\|x - x'\| / 2\theta^2)$.

The *marginal polytope* is the range of the expectation of the feature map ϕ under all distributions in a set $\mathcal{P}$, *i.e.*, $\mathcal{M} := \{\mu[p] | p \in \mathcal{P}, \mu[p] := \mathbb{E}_{x \sim p}[\phi(x)]\}$ (Wainwright & Jordan, 2003). The map $\mu : \mathcal{P} \mapsto \mathcal{H}$ associates a distribution to an element in the RKHS. For universal kernels, the elements of the marginal polytope uniquely determine distributions:

Theorem 1 (Gretton et al. (2007)) *Let k be universal and $\mathcal{P}$ denote the set of Borel probability measures p on $\mathcal{X}$ with $\mathbb{E}_{x \sim p}[k(x, x)] < \infty$. Then the map μ is injective.*

3. Kernel Moment Matching

Given a finite sample X from p, $\mu[p]$ can be approximated by the empirical mean map $\mu[X] := \frac{1}{m} \sum_{i=1}^m \phi(x_i)$. This suggests that a good estimate $\hat{p}$ of p should be chosen such that $\mu[\hat{p}]$ matches $\mu[X]$: this is the key idea of the paper. The flow of reasoning works as follows:

$$\begin{aligned} \text{density } p \rightarrow \text{ sample } X \rightarrow \text{ empirical mean } \mu[X] \\ \rightarrow \text{ density estimation via } \mu[\hat{p}] \approx \mu[X] \end{aligned} \quad (1)$$

The first line of this reasoning was established in (Altun & Smola, 2006, Theorem 15). Let $R_m(\mathcal{H}, p)$ be the Rademacher average (Bartlett & Mendelson, 2002) associated with p and $\mathcal{H}$ via

$$R_m(\mathcal{H}, p) := \tfrac{1}{m} \mathbb{E}_X \mathbb{E}_\omega \left[\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \sum_{i=1}^m \omega_i f(x_i) \right| \right].$$

where $\omega \in \{\pm 1\}$ is uniformly random. We use it to bound the deviation between empirical means and expectations:

Theorem 2 (Altun & Smola (2006)) *Assume $\|f\|_\infty \leq R$ for all $f \in \mathcal{H}$ with $\|f\|_{\mathcal{H}} \leq 1$. Then for $\epsilon > 0$ with probability at least $1 - \exp(-\epsilon^2 m R^{-2}/2)$ we have $\|\mu[p] - \mu[X]\|_{\mathcal{H}} \leq 2R_m(\mathcal{H}, p) + \epsilon$.*

This ensures that $\mu[X]$ is a good proxy for $\mu[p]$. To carry out the last step of (1) we assume the density estimator $\hat{p}$ is a mixture of a set of candidate densities p_i (or prototypes):

$$\hat{p} = \sum_{i=1}^n \alpha_i p_i \text{ where } \alpha^\top \mathbf{1} = 1 \text{ and } \alpha_i \geq 0, \quad (2)$$

where $\mathbf{1}$ is a vector of all ones. Here the goal is to obtain good estimates for the coefficients α_i and to obtain performance guarantees which specify how well $\hat{p}$ is capable of estimating p. This can be cast as an optimization problem:

$$\min_\alpha \|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}^2 \text{ s.t. } \alpha^\top \mathbf{1} = 1 \,, \alpha_i \geq 0. \quad (3)$$

To prevent overfitting, we add a regularizer $\Omega[\alpha]$, such as $\frac{1}{2}\|\alpha\|^2$, and weight it by a regularization constant $\lambda > 0$. Using the expansion of $\hat{p}$ in (2) we obtain a quadratic program (QP) for α

$$\min_\alpha \tfrac{1}{2} \alpha^\top (\mathbf{Q} + \lambda \mathbf{I}) \alpha - \mathbf{l}^\top \alpha \text{ s.t. } \alpha^\top \mathbf{1} = 1 \,, \alpha_i \geq 0, \quad (4)$$

where $\mathbf{I}$ is the identity matrix. $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{l} \in \mathbb{R}^n$ are given by

$$\mathbf{Q}_{ij} = \langle \mu[p_i], \mu[p_j] \rangle_{\mathcal{H}} = \mathbb{E}_{x \sim p_i, x' \sim p_j}[k(x, x')], \quad (5)$$

$$\mathbf{l}_j = \langle \mu[X], \mu[p_j] \rangle_{\mathcal{H}} = \tfrac{1}{m} \sum_{i=1}^m \mathbb{E}_{x \sim p_j}[k(x_i, x)]. \quad (6)$$

By construction $\mathbf{Q} \succeq 0$ is positive semidefinite, hence the program (4) is convex. We will discuss examples of kernels k and prototypes p_i where $\mathbf{Q}$ and $\mathbf{l}$ have closed form in Section 5. In many cases, the prototypes p_i also contain tunable parameters. We can also optimize them via gradient methods. Before doing so, we first explain our theoretical basis for tailoring density estimators.

4. Tailoring Density Estimators

Given functions $f \in \mathcal{H}$, a key question is to bound how well the expectations of f with respect to p can be approximated by $\hat{p}$. We have the following lemma:

Lemma 3 *Let $\epsilon > 0$ and $\epsilon' := \|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}$. Under the assumptions of Theorem 2 we have with probability at least $1 - \exp(-\epsilon^2 m R^{-2}/2)$*

$$\sup_{\|f\|_{\mathcal{H}} \leq 1} |\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim \hat{p}}[f(x)]| \leq 2R_m(\mathcal{H}, p) + \epsilon + \epsilon'.$$

Proof In the RKHS, we have $\mathbb{E}_{x \sim p}[f(x)] = \langle f, \mu[p] \rangle_{\mathcal{H}}$ and $\mathbb{E}_{x \sim \hat{p}}[f(x)] = \langle f, \mu[\hat{p}] \rangle_{\mathcal{H}}$. Hence the LHS of the bound equates to $\sup_{\|f\|_{\mathcal{H}} \leq 1} |\langle \mu[p] - \mu[\hat{p}], f \rangle|$, which is given by $\|\mu[p] - \mu[\hat{p}]\|_{\mathcal{H}}$. Using the triangle inequality, our assumption on $\mu[\hat{p}]$ and Theorem 2 completes the proof. ■

This means that we have good control over the behavior of the expectations, as long as the function class is "smooth" on $\mathcal{X}$ in terms of the Rademacher average. It also means that $\|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}$ is a sensible objective to minimize if we are only interested in approximating well the expectations over functions f.

This bound also provides the basis for tailoring density estimators. Essentially, if we have good knowledge of the function class used in an application, we can choose the corresponding RKHS or the mean map. This is equivalent to filtering the data and extracting only certain moments. Then the density estimator $\hat{p}$ can focus on matching p only up to these moments.

5. Examples

We now give concrete examples of density estimation. A number of existing methods are special cases of our setting.

Discrete Prototype or Discrete Kernel The simplest case is to represent p by a convex combination of Dirac

Table 1. Expansions for $\mathbf{Q}_{ij}$ and $\mathbf{l}_j$ when using Gaussian prototypes and various kernels in combination. Let $c := \langle x_i, x_j \rangle + 1$.

Kernel	$\mathbf{Q}_{ij}$	$\mathbf{l}_j$
Linear kernel $\langle x, x' \rangle$	$\langle x_i, x_j \rangle$	$\frac{1}{m}\sum_{i=1}^m \langle x_i, x_j \rangle$
Degree 2 polynomial kernel $(\langle x, x' \rangle + 1)^2$	$c^2 + \operatorname{tr}\Sigma_i\Sigma_j + x_i^\top \Sigma_j x_i + x_j^\top \Sigma_i x_j$	$\frac{1}{m}\sum_{i=1}^m (c^2 + x_i^\top \Sigma_j x_i)$
Degree 3 polynomial kernel $(\langle x, x' \rangle + 1)^3$	$c^3 + 6x_i^\top \Sigma_i\Sigma_j x_j + 3c(\operatorname{tr}\Sigma_i\Sigma_j + x_i^\top \Sigma_j x_i + x_j^\top \Sigma_i x_j)$	$\frac{1}{m}\sum_{i=1}^m (c^3 + 3c x_i^\top \Sigma_j x_i)$
Gaussian RBF kernel $e^{-\frac{1}{2\theta^2}\|x-x'\|^2}$	$\theta^d \left\|\Sigma_i + \Sigma_j + \theta^2\mathbf{I}\right\|^{-\frac{1}{2}} e^{-\frac{1}{2}\|x_i - x_j\|^2_{(\Sigma_i+\Sigma_j+\theta^2\mathbf{I})}}$	$\frac{1}{m}\theta^d \left\|\Sigma_j + \theta^2\mathbf{I}\right\|^{-\frac{1}{2}} \sum_{i=1}^m e^{-\frac{1}{2}\|x_i - x_j\|^2_{(\Sigma_j+\theta^2\mathbf{I})}}$

measures $p_i(x) = \delta_{x_i}$. Particle filters (Doucet et al., 2001) use this choice when approximating distributions. For instance, we could choose x_i to be the set of training points. In this case $\mathbf{Q}$ defined in (5) equals the kernel matrix and $\mathbf{l}$ is the vector of empirical kernel averages:

$$\mathbf{Q}_{ij} = k(x_i, x_j) \text{ and } \mathbf{l}_j = \frac{1}{m}\sum_{i=1}^m k(x_i, x_j). \quad (7)$$

The key difference between an unweighted set as used in particle filtering and our setting is that our expansion is specifically optimized towards good estimates with respect to functions drawn from $\mathcal{H}$.

The problem of data squashing (DuMouchel et al., 1999) can likewise be seen as a special case of kernel mean matching. Here one aims to approximate a potentially large set X by a smaller set $X' = \{(x_1, \alpha_1), \ldots, (x_n, \alpha_n)\}$ of *weighted* observations. We want to discard X and only retain X' for all further processing. If $\|\mu[X] - \mu[X']\|_{\mathcal{H}}$ is small, we expect X' to be a good proxy for X.

Instead of using generic kernels k and discrete measures δ_{x_i} as prototypes for density estimation, we may reverse their roles. That is, we may pick generic densities p_i and a Dirac kernel $k(x, x') = \delta(x = x')$. Note this is only well defined for discrete domains $\mathcal{X}$.[1] In this case the mean operator simply maps a distribution into itself and we obtain $\langle \mu[p], \mu[p'] \rangle_{\mathcal{H}} = \int_{\mathcal{X}} p(x)p'(x)\mathrm{d}x$. Using (5) we have

$$\mathbf{Q}_{ij} = \int_{\mathcal{X}} p_i(x)p_j(x)\mathrm{d}x \text{ and } \mathbf{l}_j = \frac{1}{m}\sum_{i=1}^m p_j(x_i). \quad (8)$$

Gaussian Prototype In general we will neither pick discrete prototypes nor discrete kernels for density estimation. We now give explicit expressions for Gaussian prototypes

$$p_i(x) = (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\|x - x_i\|^2_{\Sigma_i}\right), \quad (9)$$

where d is the dimension of the data, $\Sigma_i \succ 0$ is a covariance matrix, and $\|x - x'\|^2_{\Sigma_i} := (x - x')^\top \Sigma_i^{-1}(x - x')$ is the squared Mahalanobis distance. When used in conjunction with different kernels, we have the expansions in Table 1.

Other Prototypes and Kernels Other combinations of kernels and prototypes also lead to closed form expansions. For instance, similar expressions also holds for a Laplacian kernel. However, this involves more tedious integrals of the form $\int e^{-\lambda(|x|+|x-a|)}\mathrm{d}x = \lambda^{-1} + e^{-\lambda|a|}$. Another example is to use indicator functions on unit intervals centered at x_i as p_i and a Gaussian RBF kernel. In this case, both $\mathbf{Q}$ and $\mathbf{l}$ can be expressed using the error function (erf).

[1] On continuous domains such a kernel does not correspond to an RKHS since the point evaluation is not continuous.

Furthermore, Jebara et al. (2004) introduced kernels on probability functions which effectively used definition (5). While they were not motivated by the connection between kernels and density estimation, their results for rich classes of densities, such as HMMs, can be used directly to compute our $\mathbf{Q}$ and $\mathbf{l}$.

6. Related Work

Our work is related to the density estimators of Vapnik & Mukherjee (2000) and Shawe-Taylor & Dolia (2007). The main difference lies in the function space chosen to measure the approximation properties. The former uses the Banach space of functions of bounded variation, while the latter uses the space $L_1(q)$, where q denotes a distribution over test functions. For spherically invariant distributions over test functions our approach and the latter approach are identical, with a key difference (to our advantage) that our optimization is a simple QP which does not require constraint sampling to make the optimization feasible.

Support Vector Density Estimation The model of Vapnik & Mukherjee (2000) can be summarized as follows: let $F[\hat{p}]$ be the cumulative distribution function of $\hat{p}$ and let $F[X]$ be its empirical counterpart. Assume $\hat{p}$ is given by (2), and that we have a regularizer $\Omega[\alpha]$ as previously discussed. In this case the Support Vector Density Estimation problem can be written as

$$\min_{\alpha \text{ feasible}} \frac{1}{m}\sum_{i=1}^m |F[\hat{p}](x_i) - F[X](x_i)| + \lambda\Omega[\alpha]. \quad (10)$$

That is, we minimize the ℓ_1 distance between the empirical and estimated cumulative distribution functions when evaluated on the set of observations X.

To integrate this into our framework we need to extend our setting from Hilbert spaces to Banach spaces. Denote by $\mathcal{B}$ a Banach space, let $\mathcal{X}$ be a domain furnished with probability measures p, p', and let $\phi : \mathcal{X} \mapsto \mathcal{B}$ be a feature map into $\mathcal{B}$. Analogously, we define the mean map $\mu : \mathcal{P} \mapsto \mathcal{B}$ as $\mu[p] := \mathbb{E}_{x\sim p(x)}[\phi(x)]$. Moreover, we define a distance between distributions p and p' via $D(p, p') := \|\mu[p] - \mu[p']\|_{\mathcal{B}}$. If we choose $\phi(x) = \left(\chi_{(-\infty,x]}(x_1), \ldots, \chi_{(-\infty,x]}(x_m)\right)^\top$ where χ is the indicator function, and use the ℓ_1^m norm on ϕ we recover SV density estimation as a special case.

Expected Deviation Estimation Shawe-Taylor & Dolia (2007) defined a distance between distributions as follows: let $\mathcal{H}$ be a set of functions on $\mathcal{X}$ and q be a probability distribution over $\mathcal{F}$. Then the distance between two distributions p and p' is given by

$$D(p, p') := \mathbb{E}_{f\sim q(f)} \left| \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim p'}[f(x)] \right|. \quad (11)$$

That is, we compute the average distance between p and p' with respect to a distribution of test functions.

Lemma 4 *Let $\mathcal{H}$ be a reproducing kernel Hilbert space, $f \in \mathcal{H}$, and assume $q(f) = q(\|f\|_{\mathcal{H}})$ with finite $\mathbb{E}_{f\sim q}[\|f\|_{\mathcal{H}}]$. Then $D(p, p') = C \|\mu[p] - \mu[p']\|_{\mathcal{H}}$ for some constant C which depends only on $\mathcal{H}$ and q.*

Proof Note that by definition $\mathbb{E}_{x\sim p}[f(x)] = \langle \mu[p], f\rangle_{\mathcal{H}}$. Using linearity of the inner product, Equation (11) equals

$$\int |\langle \mu[p] - \mu[p'], f\rangle_{\mathcal{H}}| \, \mathrm{d}q(f)$$

$$= \|\mu[p] - \mu[p']\|_{\mathcal{H}} \int \left| \left\langle \frac{\mu[p] - \mu[p']}{\|\mu[p] - \mu[p']\|_{\mathcal{H}}}, f \right\rangle_{\mathcal{H}} \right| \mathrm{d}q(f),$$

where the integral is independent of p, p'. To see this, note that for any p, p', $\frac{\mu[p]-\mu[p']}{\|\mu[p]-\mu[p']\|_{\mathcal{H}}}$ is a unit vector which can be turned into, say, the first canonical basis vector by a rotation which leaves the integral invariant, bearing in mind that q is rotation invariant. ■

The above result covers a large number of interesting function classes. To go beyond Hilbert spaces, let $\phi : \mathcal{X} \mapsto \mathcal{B}$ be the transformation from x into $f(x)$ for all $f \in \mathcal{H}$ and $\|z\|_{\mathcal{B}} := \mathbb{E}_{f\sim q(f)}[|z_f|]$ be the $L_1(q)$ norm. Then (11) can also be written as $\|\mu[p] - \mu[p']\|_{\mathcal{B}}$, where μ is the mean map into Banach spaces. Its main drawback is the nontrivial computation for constraint sampling (de Farias & Roy, 2004) and the additional uniform convergence reasoning required. In Hilbert spaces no such operations are needed.

7. Experiments

We focus on two aspects: first, our method performs well as a density estimator *per se*; and second, it can be tailored towards the expectation over a particular function class.

7.1. Methods for Comparison

Gaussian Mixture Model (GMM)[2] The density was represented as a convex sum of Gaussians. GMM was initialized with k-means clustering. The centers, covariances and mixing proportions of the Gaussians were optimized using the EM algorithm. We used diagonal covariances in all our experiments. We always employed 50 random restarts for k-means, and returned the results from the best restart.

Parzen Windows (PZ) The density was represented as an average of a set of normalized RBF functions, with each centered on a data point. The bandwidths of the RBF functions were identical and tuned via the likelihood using leave-one-out cross validation.

Reduced Set Density Estimation (RSDE)[3] Girolami & He (2003) compressed a Parzen window estimator using RBF functions of larger bandwidths. The reduced representation was produced by minimizing an integrated squared distance between the two densities.

[2]GMM codes from: http://www.datalab.uci.edu/resources/gmm/

[3]PZ and RSDE from: http://ttic.uchicago.edu/∼ihler/code/

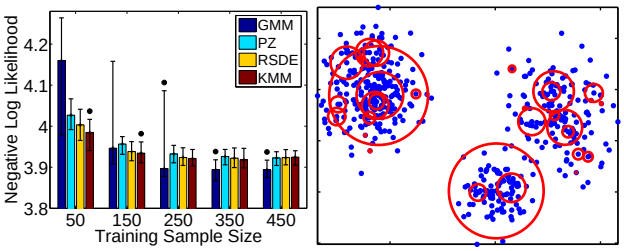

Figure 1. **Left**: Negative log-likelihood using a mixture of 3 Gaussians. The height of the bars represents the median of the scores from 100 repeats, and the whiskers correspond to the quartiles. We mark the best method with a black dot above the whiskers. **Right**: Sparsity of KMM solution. Blue dots are data points. Red circles represent prototypes selected by KMM. The size of a circle is proportional to the magnitude of its weight α_i.

Kernel Moment Matching (KMM) In applying our method, we used Gaussians with diagonal covariances as our prototypes p_i. The regularization parameter λ in our algorithm was fixed at 10^{-10} throughout. Since KMM may be tailored for different RKHS, we instantiated it with the four different kernels in Table 1. We denote them as LIN, POL2, POL3 and RBF, respectively. Our choice of kernel corresponded to the function class where we evaluated the expectations. The initialization of the prototypes will be further discussed below.

7.2. Evaluation Criterion

We compared various density estimators in terms of two criteria: negative log-likelihood and discrepancy between function expectations on test data. Since different algorithms are optimized using different criteria, we expect that each will win with respect to the criterion it employs. The benefit of our density estimator is that we can explicitly tailor for different classes of functions. For this reason, we will focus on the second criterion.

Given a function f, the discrepancy between function expectations is computed as follows: (*i*) Evaluate function expectation using test points, *i.e.*, $\frac{1}{m}\sum_{i=1}^{m} f(x_i)$; (*ii*) Evaluate function expectation using estimated density $\hat{p}$, *i.e.*, $\mathbb{E}_{x\sim\hat{p}}[f(x)]$. (*iii*) Calculate $\left|\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \mathbb{E}_{x\sim\hat{p}}[f(x)]\right|$ and normalize it by $\left|\frac{1}{m}\sum_{i=1}^{m} f(x_i)\right|$.

We will compare various methods either by repeated random instantiation or random split of the data (which we will make clear in context). For both cases, we will perform paired sign tests at the significance level of 0.01 on the results obtained from the randomizations. We will always present the median of the results in subsequent tables, and highlight in boldface those statistically equivalent methods that are significantly better than the rest.

7.3. Synthetic Dataset

In this experiment, we use synthetic datasets to compare various methods as the sample size changes. We also show

Table 2. Sparsity of the solution of RSDE and KMM. We show the median of the number of retained prototypes from 100 random initializations. Also shown are median percentages of retained prototypes after optimization.

Sample Size	50	150	250	350	450
RSDE	12 (25.0%)	35 (23.3%)	62 (24.8%)	90 (25.9%)	124 (27.5%)
KMM	**10 (20.0%)**	**21 (14.0%)**	**30 (12.0%)**	**36 (10.3%)**	**42 (9.3%)**

that KMM leads to sparse solutions.

Data Generation We generated 2 dimensional mixtures of 3 Gaussians with centers $(0,0)^\top$, $(3,3)^\top$ and $(-6,4)^\top$, and covariances $0.8^2\mathbf{I}$, $1.2^2\mathbf{I}$ and $\mathbf{I}$ respectively. The mixing proportions were 0.2, 0.3 and 0.5 respectively. We varied the training sample size while always testing on samples of size 1000. For each fixed training size, we randomly instantiated the experiments 100 times.

Experimental Protocol All training data points were used as prototypes for RSDE and KMM. Their initial covariances were set to be identical, and were initialized in both cases using the approach of RSDE. We used the RBF instance of KMM and set the bandwidth θ of the kernel to be the same as that for the prototypes. GMM used 3 centers.

Negative Log Likelihood The results are plotted in Figure 1. GMM performs best in general, while KMM is superior for small sample sizes. This is not surprising since we used a correct generative model of the data for GMM. When the sample size is small (less than 30 data points for each cluster), GMM is susceptible to local maxima and does not result in good estimates.

Sparsity of the Solution KMM also leads to sparse solutions (*e.g.*, Figure 1). When using all data points as candidate prototypes, KMM automatically prunes away most of them and results in a much reduced representation of the data. In terms of both likelihood and sparsity, KMM is superior to other reduction method such as RSDE (Table 2).

7.4. UCI Dataset

We used 15 UCI datasets to compare various methods based on the discrepancies between function expectations.

Data Description We only included real-valued dimensions of the data, and normalized each dimension of the data separately to zero mean and unit variance. For each dataset, we randomly shuffled the data points for 50 times. In each shuffle, we used the first half of the data for training and the remaining data for testing. In each shuffle, we randomly generated 100 functions f to evaluate the discrepancy criterion, *i.e.*, $f = \sum_{i=1}^{m_0} w_i k(x_i, \cdot)$ where $m_0 \in \mathbb{N}$ was uniformly random in $[1, m]$, $w_i \in \mathbb{R}$ was uniformly random in $[-1, 1]$, and the x_i were uniformly sampled from the test points. Thus, each method resulted in 5000 numbers for each dataset.

Experimental Protocol Both GMM and KMM used 10 prototypes and diagonal covariances, and both were initialized using k-means clustering. We used all four instances of KMM, namely LIN, POL2, POL3 and RBF, for the experiments, depending on the function class where we evaluated the expectations. When we used the RBF instance of KMM, we set the bandwidth θ of the kernel to the median of the distances between data points. Besides optimizing the mixing proportions of the prototypes of KMM, we also used conjugate gradient descent to optimize the center positions and covariances of the prototypes.

Table 3. Negative log-likelihood on test points as computed by various density estimators over randomizations.

Data	PZ	GMM	RSDE	LIN	POL2	POL3	RBF
covertype	11.48	**11.22**	14.97	11.64	208.29	45.23	62.37
ionosphere	**28.09**	36.58	56.68	29.55	69.92	68.79	46.49
sonar	**78.29**	119.16	122.35	**78.13**	129.81	92.35	112.21
australian	**3.32**	5.82	8.82	3.40	4.64	22.73	7.27
specft	43.01	**42.61**	43.16	**42.90**	231.76	87.28	105.04
wdbc	**25.17**	42.97	48.44	25.98	248.73	63.61	88.61
wine	19.68	21.17	22.95	**19.43**	70.15	47.99	48.94
satimage	**18.49**	39.10	59.88	20.18	158.31	121.27	52.41
segment	**-1.43**[a]	5.71	36.74	-1.07	154.25	128.74	28.38
vehicle	**10.98**	**11.99**	32.85	**11.34**	170.66	200.22	83.35
svmguide2	**27.85**	39.67	40.07	**27.92**	204.30	59.22	36.08
vowle	11.75	**6.24**	25.59	11.77	108.43	47.45	26.18
housing	**3.68**	7.44	15.53	3.81	16.07	90.51	39.88
bodyfat	**16.38**	20.06	21.96	**16.59**	87.23	171.53	53.33
abalone	**2.53**	2.57	10.17	2.75	19.15	21.77	16.29
mix3 100	2.42	**2.09**	**2.12**	2.55	2.49	2.43	2.41
mix3 500	1.91	1.92	1.92	1.94	1.93	1.91	**1.91**
mix3 1000	**1.86**	1.87	1.88	1.88	1.87	1.87	1.86

[a] Some numbers are negative, which is possible since unlike probability mass function, density can take values greater than 1.

Negative Log Likelihood Kernel moment matching can be a very different objective from the likelihood (Table 3). Except for the LIN instance, KMM results in much larger negative log-likelihood. This suggests that if the purpose of density estimation is to approximate the function expectations, likelihood is no longer a good criterion. We confirm this observation in our next experiment.

Discrepancy between Function Expectations We used four classes of functions corresponding to the RKHS of the LIN, POL2, POL3 and RBF instances of KMM. For nonlinear functions KMM clearly outperforms other density estimators, while for linear functions KMM has equivalent performance to PZ and GMM (Table 4). These results are not surprising, since KMM is explicitly optimized for approximating the function expectations well. Note that PZ is the second best for polynomial functions. This is reasonable since PZ retains all training points in the density, and should perform better than compressed representations such as GMM and RSDE. We also applied this new experimental protocol to the synthetic mixture of 3 Gaussians from the last section. We instantiate the synthetic data with 3 different sample sizes: 100, 500 and 1000. The results are shown in the last three rows of Table 3 and 4, which are consistent with those for UCI data. A closer view of the difference between GMM and KMM using "covertype" dataset is shown in Figure 2. We chose to compare GMM and KMM because they are initialized similarly.

As an aside, we remark that PZ and GMM also match the

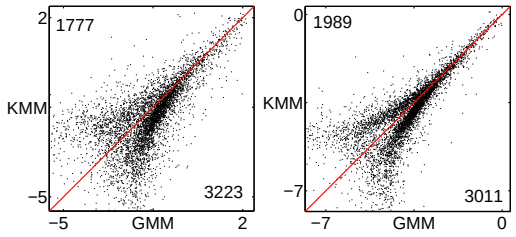

Figure 2. Scatter plot of the discrepancies between function expectations (in log scale) for the 'covertype' dataset, with GMM discrepancies on the horizontal axis and KMM discrepancies on the vertical axis. **Left**: plot for polynomial functions ($d = 2$); **Right**: plot for RBF functions. The distribution of the points is skewed below the red diagonal line, which means KMM is better. The numbers near the corners count respectively the number of points falling above and below the red diagonal.

empirical mean $\frac{1}{m}\sum_{j=1}^{m} x_j$. This is obvious for PZ. For GMM, in each EM iteration, the centers μ_i and weights α_i of each p_i are updated via $\mu_i \leftarrow \sum_{j=1}^{m} \tau_j^i x_j / \sum_{j=1}^{m} \tau_j^i$ and $\alpha_i \leftarrow \frac{1}{m}\sum_{j=1}^{m} \tau_j^i$. Here τ_j^i is the probability of x_j being generated by p_i. It follows that $\mathbb{E}_p[x] = \sum_i \alpha_i \mu_i$ also matches $\frac{1}{m}\sum_{j=1}^{m} x_j$.

8. Applications

In this section, we employ KMM for two different applications: message compression in graphical models, and image processing. The common feature of these two applications is that they involve density estimation for computing the expectation of a function, which is the relevant setting for KMM.

8.1. Message Compression

We use density estimation to compress messages in graphical models. This is of particular interest for applications such as distributed inference in sensor networks. It is our desire to compress the messages to cater for limited power supply and communication bandwidth. We will use a particle filtering example to compare GMM and KMM only, since they performed best in our earlier experiments.

We model a one dimensional time series y_t ($t = 1 \ldots 100$) as being conditionally independent given an unobserved state s_t, which is itself Markovian. This system evolves as follows:

$$s_t = f(s_{t-1}) = 1 + \sin(0.04\pi t) + 0.5 s_{t-1} + \xi \quad (12)$$

$$y_t = g(s_t) = \begin{cases} 0.2 s_t^2 + \zeta, & \text{if } t < 50, \\ 0.5 s_t - 2 + \zeta, & \text{otherwise.} \end{cases} \quad (13)$$

The random variables ξ and ζ represent process and measurement noise, respectively, and are modeled as mixtures of Gaussians,

$$\xi \sim \tfrac{1}{5}\sum_{i=1}^{5} \mathcal{N}(\mu_i, \sigma), \quad \zeta \sim \mathcal{N}(0, \sigma). \quad (14)$$

Throughout this experiment, we fix σ to 0.2 and choose μ_i to be $\{-1.5, -0.5, 0.5, 1.5, 2\}$. We initialize s_0 with ξ.

Note that our setting is a modification of de Freitas's demo[4] where we only change the process noise from a unimodal gamma distribution to a more complicated mixture model.

The task of particle filtering (Doucet et al., 2001) is to infer the hidden state given past and current observations. This can be carried out by estimating the filtering density $p(s_t|Y_t) := p(s_t|y_1, \ldots, y_t)$ recursively in a two-stage procedure. First, the current filtering density $p(s_t|Y_t)$ is propagated into the future via the transition density $p(s_{t+1}|s_t)$ to produce the prediction density $p(s_{t+1}|Y_t)$, *i.e.*,

$$\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)] := \int p(s_{t+1}|s_t) p(s_t|Y_t) \mathrm{d}s_t. \quad (15)$$

Second, $p(s_t|Y_t)$ is updated via Bayes' law,

$$p(s_{t+1}|Y_{t+1}) \propto p(y_{t+1}|s_{t+1}) p(s_{t+1}|Y_t). \quad (16)$$

The integral in (15) is usually intractable since the filtering density $p(s_t|Y_t)$ can take a complicated form. Therefore, $p(s_t|Y_t)$ is often approximated with a set of samples called particles. For distributed inference, it is these samples that need to be passed around. We want to compress the samples using density estimation such that we still do well in computing $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$. In our example, $p(s_{t+1}|s_t)$ takes the form

$$p(s_{t+1}|s_t) \propto \sum_{i=1}^{5} \exp\left(-\tfrac{(s_{t+1} - f(s_t) - \mu_i)^2}{2\sigma^2}\right). \quad (17)$$

In terms of variable s_t, $p(s_{t+1}|s_t)$ is in the RKHS with kernel $k(x, x') = \exp\left(-\frac{(x-x')^2}{2(2\sigma)^2}\right)$. We can customize KMM using this kernel, and compress messages by targeting a good approximation of $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$.

We use 5 centers for both GMM and KMM to compress the messages. We compare the filtering results with the true states. The error is measured as the root mean square of the deviations. The results for compressing different numbers of particles are reported in Table 5. We find that filtering results after compression even outperform those obtained from the full set of particles (PF). In particular, the results for KMM are slightly better than those for GMM. By compression, we have extracted the information most essential to statistical inference, and actually made the inference more robust. If the compression is targeted to $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$ (as we do in KMM), we can simply get better results.

The shortcomings of general purpose density estimation also arise in the more general settings of message passing and belief propagation. This is due to the way messages are constructed: given a clique, the incoming messages are multiplied by the clique potential and all variables not in the receiver are integrated out. In most cases, this makes the outgoing messages very complicated, causing significant computational problems. Popular methods include

[4] http://www.cs.ubc.ca/∼nando/software/upf_demos.tar.gz

Table 4. Discrepancy between function expectations over randomizations. Smaller numbers are not necessarily statistical significant.

Data	Linear Functions				Polynomials ($d=2$)				Polynomials ($d=3$)				RBF Functions			
	PZ	GMM	RSDE	LIN	PZ	GMM	RSDE	POL2	PZ	GMM	RSDE	POL3	PZ	GMM	RSDE	RBF
covertype	**2.003**	**2.003**	10.280	**2.003**	0.185	0.194	0.396	**0.150**	**0.418**	0.539	1.240	0.412	0.073	0.023	0.071	**0.020**
ionosphere	**2.006**	**2.006**	17.995	**2.006**	**0.159**	0.232	0.383	0.169	**0.615**	0.664	1.659	0.626	0.120	0.024	0.142	**0.022**
sonar	**2.000**	**2.000**	12.288	**2.000**	0.971	0.354	0.933	**0.242**	0.691	0.745	2.558	**0.673**	0.857	0.030	0.873	**0.029**
australian	**2.000**	**2.000**	14.217	**2.000**	**0.369**	0.380	0.587	0.380	**0.832**	0.837	1.031	0.833	0.089	0.028	0.106	**0.024**
specft	**2.000**	**2.000**	3.594	**2.000**	0.891	0.515	0.522	**0.488**	0.922	0.878	1.265	**0.867**	0.903	0.067	0.904	**0.062**
wdbc	**2.004**	**2.004**	16.447	**2.004**	0.209	0.233	0.406	**0.166**	0.519	0.612	1.362	**0.512**	0.482	0.027	0.456	**0.023**
wine	**2.017**	**2.017**	9.489	**2.017**	0.822	0.236	1.027	**0.211**	**0.679**	0.718	2.782	**0.682**	0.471	0.040	0.545	**0.039**
satimage	**2.000**	**2.000**	27.561	**2.000**	0.146	0.126	0.533	**0.122**	0.260	0.281	1.230	**0.256**	0.307	0.028	0.359	**0.026**
segment	**2.003**	**2.003**	23.388	**2.003**	**0.258**	0.245	0.803	**0.263**	**0.590**	0.572	1.021	**0.588**	0.053	0.025	0.247	**0.022**
vehicle	**2.005**	**2.005**	26.331	**2.005**	**0.126**	0.135	0.780	**0.119**	**0.496**	**0.478**	1.686	**0.493**	0.095	0.028	0.325	**0.027**
svmguide2	**2.005**	**2.005**	7.248	**2.005**	3.468	0.247	3.341	**0.183**	0.866	0.782	2.603	**0.729**	0.798	0.019	0.808	**0.018**
vowle	**2.000**	**2.000**	12.913	**2.000**	**0.131**	0.150	0.642	**0.131**	**0.348**	0.394	1.741	**0.352**	0.028	0.019	0.111	**0.018**
housing	**2.000**	**2.000**	7.668	**2.000**	**0.117**	0.126	0.399	**0.121**	**0.393**	0.421	0.890	**0.391**	0.044	0.027	0.091	**0.025**
bodyfat	**2.000**	**2.000**	7.295	**2.000**	0.288	0.243	0.595	**0.242**	1.029	**1.017**	1.200	**1.015**	0.430	0.038	0.432	**0.037**
abalone	**2.005**	**2.005**	17.010	**2.005**	0.105	0.101	0.234	**0.103**	0.629	0.636	3.308	**0.628**	0.049	0.044	0.294	**0.043**
mix3 100	**2.000**	**2.000**	2.164	**2.000**	**0.153**	**0.152**	0.164	**0.152**	0.248	**0.242**	0.271	0.244	0.046	**0.044**	0.046	**0.044**
mix3 500	**2.000**	**2.000**	2.069	**2.000**	0.064	0.062	0.064	**0.062**	0.094	**0.092**	0.097	**0.091**	0.020	0.019	0.020	**0.019**
mix3 1000	**2.000**	**2.000**	2.035	**2.000**	0.052	**0.051**	0.051	**0.050**	0.082	0.081	0.082	**0.080**	0.015	**0.014**	**0.015**	**0.014**

Table 5. Root mean square error and standard deviation of the filtering results before and after particle compression. We randomly instantiated the system 50 times and concatenate the times to produce the results. Statistical tests are done by viewing each time point as a data point.

Particle #	PF	GMM	KMM
100	0.683±0.114	**0.558±0.084**	**0.546±0.072**
500	0.679±0.111	**0.556±0.076**	**0.530±0.070**
1000	0.685±0.111	0.556±0.082	**0.526±0.070**

particle filtering, which uses a discrete approximation of the messages, and expectation propagation, which uses a single Gaussian approximation of the messages (Minka, 2001). We plan to further investigate KMM in these general settings. Our key benefit is that we can customize the approximation properties for a particular graphical model.

8.2. Image Retrieval and Categorization

Following the work of (Rubner et al., 2000; Greenspan et al., 2002), we use density estimation as an intermediate step for image retrieval and categorization.

8.2.1. Image Retrieval

Image retrieval is the task of finding from a given database the set of images similar to a given query image. An image is normally characterized by the distribution over features (*e.g.*, color, texture) of pixels, patches, etc. It is thus helpful to compress the distribution by density estimation into more compact forms (*e.g.*, mixtures of Gaussians), on which the query is based. In particular, the advantage is that density estimation can be computed offline before the query takes place, thus offering computational and storage savings.

Method Greenspan et al. (2002) used GMM for density estimation; we propose KMM as an alternative. After density estimation, the dissimilarity between two distributions needs to be measured and the Earth Mover's Distance (EMD) is a state-of-the-art measure. Given two distributions represented by sets of weighted prototypes, EMD regards one collection as mass of earth spread in the feature space, while the other is a collection of holes. The EMD is defined as the least amount of work needed to fill the holes with earth. A unit of work corresponds to the ground distance between two prototypes. If we represent the distributions by mixtures of Gaussians, then a sensible ground distance $D(p_i, p'_j)$ between two Gaussians $p_i = \mathcal{N}(\mu_i, \Sigma_i)$ and $p'_j = \mathcal{N}(\mu'_j, \Sigma'_j)$ is the Fréchet distance used in (Greenspan et al., 2002),

$$D^2(p_i, p'_j) := |\mu_i - \mu'_j|^2 + \mathrm{tr}\left(\Sigma_i + \Sigma'_j - 2(\Sigma_i\Sigma'_j)^{1/2}\right).$$

Based on $D(p_i, p'_j)$, if $p = \sum_i \alpha_i p_i$ where p_i is a Gaussian and α_i is its weight, and similarly $p' = \sum_j \alpha'_j p'_j$, then the EMD between p and p' is

$$\mathrm{EMD}(p, p') := \min_{\gamma_{ij} \text{ feasible}} \sum_i \sum_j \gamma_{ij} D(p_i, p'_j),$$

where $\gamma_{ij} \geq 0$ is the flow between p_i and q_j. Feasibility means $\sum_i \gamma_{ij} \leq \alpha'_j$ and $\sum_j \gamma_{ij} \leq \alpha_i$ for all i and j.

Settings In this experiment, the distance measure is fixed to EMD. We plug the densities estimated by GMM and KMM into EMD[5], and compare the retrieval results. Parameters for KMM and GMM were chosen in the same way as in Section 7.4. Here KMM used POL3. For each image, we sampled 10^3 pixels and each pixel's feature vector was the CIE-Lab value of a 5×5 window centered on it.

Results We collected $L = 10537$ images from various sources including FIRE and CIRES[6]. The dataset included 10 labeled categories like horse, beach, and each category has 100 images. For each image $I_c(i)$ from class c ($c \in \{1, ..., 10\}$, $i \in \{1, ..., 100\}$), we retrieved r ($r \in \{1, ..., L\}$) closest images (in terms of EMD) from the whole database and counted how many among them are also from class c, denoted as $g_c(i, r)$ for GMM and $k_c(i, r)$ for KMM. For each c and r, we performed a paired sign test between $\{g_c(i, r)\}_{i=1}^{100}$ and $\{k_c(i, r)\}_{i=1}^{100}$. Since p-value is always in (0,1], we report in Figure 3 the log p-value if the median of $\{k_c(i, r) - g_c(i, r)\}_{i=1}^{100}$ is higher than 0. Otherwise, we plot the *negative* log p-value. Negative values are in favor of KMM. In Figure 3, performance of KMM is superior to or competitive with GMM in 8 categories and for most values of r (number of retrieved images).

[5]EMD code from http://ai.stanford.edu/∼rubner/emd

[6]FIRE: http://www-i6.informatik.rwth-aachen.de/∼deselaers/fire.html, CIRES: http://cires.matthewriley.com

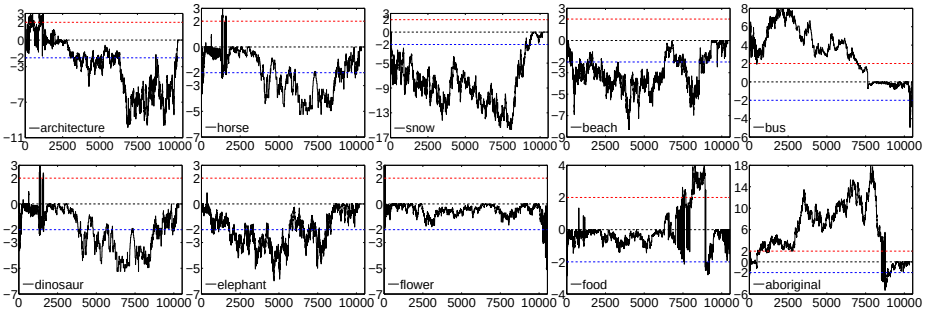

Figure 3. Log sign test p-value (vertical axis) v.s. # retrieved images (horizontal axis). Negative if KMM is better than GMM, and positive otherwise. ± 2 for significance level 0.01.

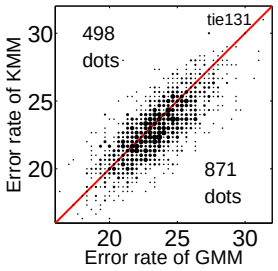

Figure 4. Error rate of image categorization using KMM and GMM.

It is important to note that the Fréchet distance is *not* in the class of functions used in KMM, and KMM still performs reasonably well. In the next section, we learn an image classifier using the *same* kernel as used in KMM.

8.2.2. Image Categorization

A closely related but different task is learning to categorize images using multi-class classification, particularly by SVM. Here all we need is a kernel between pairs of image densities p and q, which is readily given by $\langle \mu[p], \mu[q] \rangle_{\mathcal{H}}$. The SVM classifier takes the form $f(p_j) = \sum_i \gamma_i \langle \mu[p_i], \mu[p_j] \rangle = \mathbb{E}_{x \sim p_j}[\sum_i \gamma_i \mu[p_i](x)]$ for some $\gamma_i \in \mathbb{R}$. Since $\sum_i \gamma_i \mu[p_i] \in \mathcal{H}$, KMM ensures that p_j is estimated such that this expectation is well approximated.

Our 10-class classification used 1000 images from the 10 categories. We randomly divided each category into 70 images for training and 30 images for testing. We used LibSVM to train a multi-class SVM with one-against-one criterion on the combined 700 training images. The loss and regularization tradeoff parameter was determined by an inner loop 10-fold cross validation on the training data. Finally we test the accuracy of the learned model on the 300 test images. The whole process is repeated for 1500 times. We use POL3 for both KMM and SVM, because for both GMM and KMM, POL3 significantly outperforms POL2 and RBF in practice[7]. By using paired sign test, KMM yields lower error rate than GMM at significance level 0.01. Figure 4 shows the scatter plot of the resulting error rates.

Acknowledgements NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. This work is also supported by the IST Program of the European Community, under the FP7 Network of Excellence, ICT-216886-NOE.

References

Altun, Y., & Smola, A. (2006). Unifying divergence minimization and statistical inference via convex duality. In *COLT 2006*.

Barndorff-Nielsen, O. E. (1978). *Information and Exponential Families in Statistical Theory*.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, *3*, 463–482.

de Farias, N., & Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, *29*(3).

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Dudík, M., Phillips, S., & Schapire, R. (2004). Performance guarantees for regularized maximum entropy density estimation. In *COLT 2004*.

DuMouchel, W., Volinsky, C., Cortes, C., Pregibon, D., & Johnson, T. (1999). Squashing flat files flatter. In *KDD 1999*.

Girolami, M., & He, C. (2003). Probability density estimation from optimally condensed data samples. *IEEE TPAMI*, *25*(10), 1253–1264.

Greenspan, H., Dvir, G., & Rubner, Y. (2002). Context-based image modeling. In *ICPR 2002*.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., & Smola, A. (2007). A kernel method for the two-sample-problem. In *NIPS 2007*.

Jebara, T., Kondor, R., & Howard, A. (2004). Probability product kernels. *JMLR*, *5*, 819–844.

McLachlan, G., & Basford, K. (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker.

Minka, T. (2001). *Expectation Propagation for approximate Bayesian inference*. Ph.D. thesis, MIT.

Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Stat.*, *33*, 1065–1076.

Rubner, Y., Tomasi, C., & Guibas, L. (2000). The earth mover's distance as a metric for image retrieval. *Intl. J. Computer Vision*, *40*(2), 99–121.

Shawe-Taylor, J., & Dolia, A. (2007). A framework for probability density estimation. In *AISTATS 2007*.

Silverman, B. W. (1986). *Density estimation for statistical and data analysis*. Monographs on statistics and applied probability. Chapman and Hall.

Smola, A., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *ALT 2007*.

Steinwart, I. (2002). The influence of the kernel on the consistency of support vector machines. *JMLR*, *2*, 463–482.

Vapnik, V., & Mukherjee, S. (2000). Support vector method for multivariate density estimation. In *NIPS 12*.

Wainwright, M. J., & Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. Tech. Rep. 649, UC Berkeley.

[7] The error rate of GMM using POL3 is $23.3 \pm 2.24\%$, outperforming POL2 ($23.8 \pm 2.14\%$) at significance level 0.01. KMM has $22.9 \pm 2.19\%$ error using POL3, beating POL2 ($24.3 \pm 2.14\%$) at significance level 0.01. Using an RBF kernel where the fixed value of the bandwidth θ was tested over $0.01, 0.1, 1, 10$, and 100 times the median distance between 1000 images, both GMM and KMM incur over 50% error.