

Mixture Density Mercer Kernels: A Method to Learn Kernels Directly from Data*

Ashok N. Srivastava, Ph.D.[†]

Abstract

This paper presents a method of generating Mercer Kernels from an ensemble of probabilistic mixture models, where each mixture model is generated from a Bayesian mixture density estimate. We show how to convert the ensemble estimates into a Mercer Kernel, describe the properties of this new kernel function, and give examples of the performance of this kernel on unsupervised clustering of synthetic data and also in the domain of unsupervised multispectral image understanding.

Keywords: Clustering, Mixture Density Estimation, Kernel Methods, Unsupervised Learning, Image Segmentation

1 Introduction and Previous Work

This paper addresses the problem of estimating the posterior probability of a continuous random variable $\mathbf{x} \in \mathcal{R}^d$ assuming that the underlying data generating process is possibly a Gaussian mixture density function. The posterior probability we wish to estimate is $P(c|\mathbf{x}_i)$ where \mathbf{x}_i is one of N independent and identically distributed realizations from a data set \mathcal{X} and c is one of C clusters or modes. There is an abundant literature on mixture density estimation, as it arises in classical statistics [13], statistical machine learning [9], and data mining [4]. Recently, there has been additional work in performing clustering [8] or density estimation [12] in high, possibly infinite dimensional Hilbert spaces defined by Mercer Kernels.

Traditional algorithms used to generate the probability density $P(\mathbf{x}|\Theta)$ assume that:

$$(1.1) \quad P(\mathbf{x}|\Theta) = \sum_{c=1}^C P(c)P(\mathbf{x}|\theta_c)$$

where Θ is a vector containing the C model parameters, and θ_c are the model parameters for the c th mixture component. The parameters of such a model are obtained through Expectation Maximization of the ap-

propriate log-likelihood function or, more generally, the posterior log-likelihood.

We propose to assess the posterior class probabilities by generating a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that measures the similarity between two data points \mathbf{x}_i and \mathbf{x}_j through the use of an ensemble of mixture densities of the form given above. We show that under certain simplifying conditions this kernel function, which we call a *mixture density kernel* is a Gram matrix that measures the number of times on average an ensemble of mixture density estimates agree that two points arise from the same mode of the probability density function. This kernel function capitalizes on the fact that each estimated probability density function is a non-optimal sample from a model space, and relies heavily on the theory of bagged classifiers [2]. We also show that the ensemble of mixture models is a method to empirically combine the posterior probabilities, and relations to the theory of kernel alignment are given. Due to the fact that the kernel function can be derived from an ensemble of Bayesian mixture models, we have the ability to encode domain knowledge in the kernel function through the use of informative priors, thus aiding the search for an optimal model. Although this paper presents results of this kernel function on unsupervised learning problems, they can be applied equally well to typical supervised learning problems such as classification and regression in a straightforward manner.

The paper begins with a description of the notation used in this paper, gives a brief introduction to Bayesian mixture modelling, introduces Mercer kernels and shows how they can be interpreted as a similarity measure, introduces Mixture Density Kernels and compares them with other kernel functions, and then gives experimental results of the algorithm on unsupervised learning in a synthetic data set and then a real-world task of multispectral image understanding.

2 Notation

- p is the dimension of the data
- \mathcal{M} is the space of models from which the mixture density models are drawn.

*This work is funded in part by NASA's Intelligent Data Understanding program.

[†]NASA Ames Research Center

- \mathcal{F} is the feature space, which may be a high dimensional (but finite) space, or more generally an infinite dimensional Hilbert space.
- N is the number of data points \mathbf{x}_i drawn from a p dimensional space
- M is the number of probabilistic models used in generating the kernel function.
- C is the number of mixture components in each probabilistic model. In principle one can use a different number of mixture components in each model. However, here we choose a fixed number for simplicity.
- \mathbf{x}_i is a $p \times 1$ dimensional real column vector that represents the data sampled from a data set \mathcal{X} .
- $\Phi(\mathbf{x}) : \mathcal{R}^p \mapsto \mathcal{F}$ is generally a nonlinear mapping to a high, possibly infinite dimensional, feature space \mathcal{F} . This mapping operator may be explicitly defined or may be implicitly defined via a kernel function.
- $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_j) \in \mathcal{R}$ is the kernel function that measures the similarity between data points \mathbf{x}_i and \mathbf{x}_j . If K is a Mercer kernel, it can be written as the outer product of the map Φ . As i and j sweep through the N data points, it generates an $N \times N$ kernel matrix.
- Θ is the entire set of parameters that specify a mixture model.

3 Mixture Models: A Sample from a Model Space

In this section, we briefly motivate the development of Mixture Density Kernels by showing that the combined result of model misspecification and the effects of a finite data set can lead to high uncertainty in the estimate of a mixture model. We closely follow the arguments given in [16].

Suppose that a data set \mathcal{X} is generated by drawing a finite sample from a mixture density function $f(\Lambda^*, \Theta^*)$, where Λ^* defines the true density function (say a Gaussian mixture density) and is a sample from a large but finite class of models \mathcal{M} , and Θ^* defines the true set

of parameters of that density function. In the case of the Gaussian mixture density, these parameters would be the means and covariance matrices for each Gaussian component, and the number of components that comprise the model. We can compute the probability of obtaining the correct model given the data as follows (see Smyth and Wolpert, 1998 for a detailed discussion). The posterior probability of the true density $f^* \equiv f(\Lambda^*, \Theta^*)$ given the data \mathcal{X} is:

$$P(f(\Lambda^*, \Theta^*)|\mathcal{X}) = \int_{\mathcal{M}} \int_{\mathcal{R}(\mathcal{M})} P(\Lambda, \Theta|\mathcal{X}) \times \delta(f^* - P(\Lambda, \Theta)) d\Lambda d\Theta$$

where the first integral is taken over the model space \mathcal{M} and the second integral is taken over the region in the parameter space that is appropriate for the model Λ , and δ is the Dirac delta function. Using Bayes rule, it is possible to expand the posterior into a product of the posterior of the model uncertainty and the posterior of the parameter uncertainty. Thus, we have:

$$\begin{aligned} P(f^*|\mathcal{X}) &= \int_{\mathcal{M}} \int_{\mathcal{R}(\mathcal{M})} P(\Lambda|\mathcal{X}) P(\Theta|\Lambda, \mathcal{X}) \times \\ &\quad \delta(f^* - P(\Lambda, \Theta)) d\Lambda d\Theta \\ &= \frac{1}{P(\mathcal{X})} \int_{\mathcal{M}} \int_{\mathcal{R}(\mathcal{M})} P(\Theta|\Lambda, \mathcal{X}) P(\Lambda, \mathcal{X}) \times \\ &\quad \delta(f^* - P(\Lambda, \Theta)) d\Lambda d\Theta \end{aligned}$$

The first equation above shows that there are two sources of variation in the estimation of the density function. The first is due to model misspecification, and the second is due to parameter uncertainty. The second equation shows that if prior information is available, it can be used to modify the likelihood of the data in order to obtain a better estimate of the true density function. The goal of this paper is to seek a representation of the posterior $P(\mathbf{x}_i|\Theta)$ by reducing these errors by embedding \mathbf{x}_i in a high dimensional feature space that defines a kernel function.

4 Review of Kernel Functions

Mercer Kernel functions can be viewed as a measure of the similarity between two data points that are embedded in a high, possibly infinite dimensional feature space. For a finite sample of data \mathcal{X} , the kernel function yields a symmetric $N \times N$ positive definite matrix, where the (i, j) entry corresponds to the similarity between $(\mathbf{x}_i, \mathbf{x}_j)$ as measured by the kernel function. Because of the positive definite property, such a Mercer Kernel can be written as the outer product of the data in the feature space. Thus, if $\Phi(\mathbf{x}_i) : \mathcal{R}^d \mapsto \mathcal{F}$ is the (perhaps implicitly) defined embedding function,

we have $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_j)$. Typical kernel functions include the Gaussian kernel for which $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, and the polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^p$.

For supervised learning tasks, linear algorithms are used to define relationships between the target variable and the embedded features [6]. Work has also been done in using kernel methods for unsupervised learning tasks, such as clustering [8, 17] and density estimation [12].

5 Mixture Density Kernels

The idea of using probabilistic kernels was discussed by Haussler in 1999 [10] where he observes that if $K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{X} \times \mathcal{X}$, and $\sum_{\mathbf{x}_i} \sum_{\mathbf{x}_j} K(\mathbf{x}_i, \mathbf{x}_j) = 1$ then K is a probability distribution and is called a P-Kernel. He further observed that the Gibbs kernel $K(\mathbf{x}_i, \mathbf{x}_j) = P(\mathbf{x}_i)P(\mathbf{x}_j)$ is also an admissible kernel function.

Although these kernel functions either represent or are derived from probabilistic models, they may not measure similarity in a consistent way. For example, suppose that \mathbf{x}_i is a low probability point, so that $P(\mathbf{x}_i) \approx 0$. In the case of the Gibbs kernel, $K(\mathbf{x}_i, \mathbf{x}_i) \approx 0$, although the input vectors are identical. The Gram matrix generated by this kernel function would only show those points as being similar which have very high probabilities. While this feature may be of value in some applications, it needs modification to work as a similarity measure.

Our idea is to use an ensemble of probabilistic mixture models as a similarity measure. Two data points will have a larger similarity if multiple models agree that they should be placed in the same cluster or mode of the distribution. Those points where there is disagreement will be given intermediate similarity measures. The shapes of the underlying mixture distributions can significantly affect the similarity measurement of the two points. Experimental results uphold this intuition and show that in regions where there is “no question” about the membership of two points, the Mixture Density Kernel behaves identically to a standard mixture model. However, in regions of the input space where there is disagreement about the membership of two points, the behavior may be quite different than the standard model. Since each mixture density model in the ensemble can be encoded with domain knowledge by constructing informative priors, the Bagged Probabilistic Kernel will also encode domain knowledge. The Bagged Probabilistic Kernel is defined as follows:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) \\ &= \frac{1}{Z(\mathbf{x}_i, \mathbf{x}_j)} \sum_{m=1}^M \sum_{c_m=1}^{C_m} P_m(c_m|\mathbf{x}_i)P_m(c_m|\mathbf{x}_j) \end{aligned}$$

The feature space is thus defined explicitly as follows:

$$\begin{aligned} \Phi(\mathbf{x}_i) &\propto [P_1(c=1|\mathbf{x}_i), P_1(c=2|\mathbf{x}_i), \dots, \\ &P_1(c=C|\mathbf{x}_i), P_2(c=1|\mathbf{x}_i), \dots, P_M(c=C|\mathbf{x}_i)] \end{aligned}$$

The first sum in the defining equation above sweeps through the M models in the ensemble, where each mixture model is a Maximum A Posteriori estimator of the underlying density trained by sampling (with replacement) the original data. We will discuss how to design these estimators in the next section. C_m defines the number of mixtures in the m th ensemble, and c_m is the cluster (or mode) label assigned by the model. The quantity $Z(\mathbf{x}_i, \mathbf{x}_j)$ is a normalization such that $K(\mathbf{x}_i, \mathbf{x}_i) = 1$ for all i . The fact that the Mixture Density Kernel is a valid kernel function arises directly from the definition. In order to prove K is a valid kernel function, we need to show that it is symmetric, and that the kernel matrix is positive semi-definite[6]. The kernel is clearly symmetric since $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$ for all values of i and j . The proof that K is positive semi-definite is straightforward and arises from the fact that Φ can be explicitly known. For nonzero α and a Gram matrix K formed by choosing an arbitrary input set $\{\mathbf{x}_i\}_{i=1}^p$:

$$(5.2) \quad \alpha^T K \alpha = \alpha^T \Phi \Phi^T \alpha = \beta^T \beta \geq 0$$

The Mixture Density Kernel function can be interpreted as follows. Suppose that we have a hard classification strategy, where each data point is assigned to the most likely posterior class distribution. In this case the kernel function counts the the number of times the M mixtures agree that two points should be placed in the same cluster mode. In soft classification, two data points are given an intermediate level of similarity (between 0 and 1) which will be less than or equal to the case where all models agree on their membership, in which case the entry would be unity. Further interpretation of the kernel function is possible by applying Bayes rule to the defining equation of the Mixture Density Kernel. Thus, we have:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{Z(\mathbf{x}_i, \mathbf{x}_j)} \sum_{m=1}^M \sum_{c_m=1}^{C_m} \frac{P_m(\mathbf{x}_i|c_m)P_m(c_m)}{P_m(\mathbf{x}_i)} \times \\ &\quad \frac{P_m(\mathbf{x}_j|c_m)P_m(c_m)}{P_m(\mathbf{x}_j)} \\ &= \frac{1}{Z(\mathbf{x}_i, \mathbf{x}_j)} \sum_{m=1}^M \sum_{c_m=1}^{C_m} \frac{P_m(\mathbf{x}_i, \mathbf{x}_j|c_m)P_m^2(c_m)}{P_m(\mathbf{x}_i, \mathbf{x}_j)} \end{aligned}$$

The second step above is valid under the assumption that the two data points are independent and identically

distributed. This equation shows that the Mixture Density Kernel measures the ratio of the probability that two points arise from the same mode, compared with the unconditional joint distribution. If we simplify this equation further by assuming that the class distributions are uniform, the kernel tells us on average (across ensembles) the amount of information gained by knowing that two points are drawn from the same mode in a mixture density.

It is not possible to obtain similarity through computing the average density across an ensemble (which would be the traditional approach to bagging), because in that case, $P(x) = \sum_{m=1}^M \sum_{k=1}^K P_m(k|x)P_m(k)$ the assignment of a data point to a given mixture component is arbitrary. Thus, for two given probabilistic models P_m and P_n , cluster c_m may not be the same as cluster c_n . This lack of similarity may go beyond the mere problem that the mixture assignments are arbitrary. The geometry of the mixture components may be different from run to run. For example, in the case of a Gaussian mixture model, the positions of the mean vectors μ_m and μ_n and their associated covariance matrices may be different.

The dimension of the feature space defined by the Mixture Density Mercer Kernel is large but finite. In the case where the number of components varies for each member, the dimensionality is $\dim(\mathcal{F}) = \sum_{m=1}^M C_m$. Once normalized to unit length (through the factor $Z(\mathbf{x}_i, \mathbf{x}_j)$), the $\Phi(\mathbf{x}_i)$ vectors represent points on a high dimensional hypersphere. The angle between the vectors determines the similarity between the two data points, and the (i, j) element of the kernel matrix is the cosine of this angle. We take this opportunity to point out one important aspect of this kernel function. While it is clear that the kernel as we have defined it obeys the properties of a Mercer Kernel, it should be noted that most Mercer Kernels do not calculate the embedding space directly. Rather, they are implicitly defined, thus reducing the computational requirement.

This kernel function is similar to the Cluster-based Similarity Partitioning Algorithm (CSPA) discussed in [1]. The implementation that they discuss uses a hard partitioning, however the also mention that the approach can be extended to soft clusterings by using the posterior probabilities of cluster membership, as is done in this work. The similarity matrix can be used in further clustering algorithms.

Building the Mixture Density Kernel requires building an ensemble of mixture density models, each representing a sample from the larger model space \mathcal{M} . The greater the heterogeneity of the models used in generating the kernel, the more effective the procedure. In our implementation of the procedure, the training data

is sampled M times with replacement. These overlapping data sets, combined with random initial conditions for the EM algorithm, aid in generating a heterogenous ensemble. Other ways of introducing heterogeneity include encoding domain knowledge in the model. This can be accomplished through the use of Bayesian Mixture Densities, which is the subject of the next section.

6 Review of Bayesian Mixture Density Estimation

A Bayesian formulation to the density estimation problem requires that we maximize the posterior distribution $P(\Theta|\mathcal{X})$ which arises as follows:

$$(6.3) \quad P(\Theta|\mathcal{X}) = \frac{P(\mathcal{X}|\Theta)P(\Theta)}{P(\mathcal{X})}$$

Cheeseman and Stutz 1995, among others, showed that prior knowledge about the data generating process can be encoded in the prior $P(\Theta)$ in order to guide the optimization algorithm toward a model Θ' that takes the domain knowledge into account. This prior assumes that a generative model Λ has been chosen (such as a Gaussian), and determines the prior over the model parameters.

A Bayesian formulation to the mixture density problem requires that we specify the model (Λ) and then a prior distribution of the model parameters. In the case of a Gaussian mixture density model for $\mathbf{x} \in \mathcal{R}^d$, we take the likelihood function as:

$$\begin{aligned} P(\mathbf{x}|\theta_c) &= P(\mathbf{x}|\mu_c, \Sigma_c, c) \\ &= (2\pi)^{-\frac{d}{2}} |\Sigma_c|^{-\frac{1}{2}} \times \\ &\quad \exp\left[-\frac{1}{2}(\mathbf{x} - \mu_c)^T \Sigma_c^{-1}(\mathbf{x} - \mu_c)\right] \end{aligned}$$

In the event that domain information is to be encoded, it is convenient to represent it in terms of a conjugate prior for the Gaussian distribution. A conjugate is defined as follows:

DEFINITION 6.1. *A family F of probability density functions is said to be conjugate if for every $f \in F$, the posterior $f(\Theta|\mathbf{x})$ also belongs to F .*

For a mixture of Gaussians model, priors can be set as follows[3]:

- For priors on the means, either a uniform distribution or a Gaussian distribution can be used.
- For priors on the covariance matrices, the Wishart density can be used: $P(\Sigma_i|\alpha, \beta, \mathbf{J}) \propto |\Sigma_i^{-1}|^{\frac{\alpha}{2}} \exp(-\text{atr}(\Sigma_i^{-1}\mathbf{J})/2)$.

- For priors on the mixture weights, a Dirichlet distribution can be used: $P(p_i|\gamma) \propto \prod_{c=1}^C p_i^{\gamma_i - 1}$, where $p_i \equiv P(c = i)$.

These priors can be viewed as regularizers for the mixture network as in [15]. Maximum a posteriori estimation is performed by taking the log of the posterior likelihood of each data point \mathbf{x}_i given the model Θ . The following function is thus optimized using the Expectation Maximization[7]:

$$(6.4) \quad l(\Theta) = \log \left[\prod_{i=1}^N P(\mathbf{x}_i|\Theta)P(\Theta) \right]$$

In some cases, details of the underlying distribution are known and can be used to influence the estimated distribution. For example, in some cases there may be prior knowledge of the class distribution based on previous work, in which case the Dirichlet distribution would be appropriate. Many studies can be performed using noninformative priors, in which case $P(\Theta) \equiv 1$. In the former case, the mixture density kernel takes domain information into account, whereas in the latter case, it is determined directly from the data.

7 Comparison with Other Kernels

We now qualitatively compare the Mixture Density Kernel with three other types of kernels: the Gaussian or Radial Basis Function (RBF) kernel and other parametric kernels such as the polynomial kernel, the Fisher kernel, and kernel alignment, where the functions are designed to maximize the accuracy of a prediction.

The RBF kernel and other parametric kernel functions are usually chosen using some heuristic method where the classification accuracy or other criterion is used to choose the best kernel. In the case of the Mixture Density Kernel, the underlying structure of the data is used to generate the kernel function. Thus, it can improve upon the standard set of kernel functions and can be used in situations where no clear objective criterion is available, such as in unsupervised learning problems like clustering. The parametric kernels are appropriate for use in unsupervised and supervised problems alike.

The Fisher kernel as developed by Jaakkola and Haussler (1999) [11] and the ‘‘Tangent of vectors of posterior log odds’’ [18] kernel use probabilistic models to generate kernel functions. Kernel Alignment [5] is another method to generate kernel functions, but does not use a probabilistic model as its underlying basis. However, these kernel functions are optimized for discriminative performance and may not be appropriate for unsupervised problems such as clustering.

8 Kernel Clustering in Feature Space

Girolami, (2001) has given an algorithm to perform clustering in the feature space using an approach similar to k-means clustering. A brief review of the method follows. The cost function for k-means clustering in the feature space at a given instant in time is:

$$G^\Phi = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K q_{ki} [\Phi(Z_i) - m_k^\Phi]^T \times [\Phi(Z_i) - m_k^\Phi]$$

where q_{ki} is the cluster membership indicator function ($q_{ki} = 1$ if vector Z_i is a member of cluster k , and zero otherwise), and m_k^Φ is the cluster center in the feature space. Thus, if we expand the right-hand side of the above equation, and take $m_k^\Phi = \frac{1}{N} \sum_{i=1}^N q_{ki} \Phi(Z_i)$, which represents the centroid of the cluster in feature space, we obtain an equation in which only inner products appear. The nonlinear mapping Φ does not need to be determined explicitly because the kernel function is taken as the inner product in the feature space: $K_{ij} = \Phi^T(Z_i)\Phi(Z_j)$. The objective of kernel clustering is to find a membership function q and cluster centers m^Φ that minimize the cost G^Φ . Various methods can be used to minimize G^Φ , including annealing methods (as described in Girolami, 2001) or direct search. If the annealing method is used, the cluster centers are implicitly defined. However, depending on the kernel function used, a direct search approach allows for the pre-image of the cluster center to be explicitly known. Note that K , the number of clusters in the feature space, need not be directly related to $\sum_{c=1}^M C_m$, which is the number of dimensions of the feature space, nor does K need to be a direct function of the number of modes in each mixture ensemble. After the optimization is completed, it is possible to compute the uncertainty in the clustering by computing the entropy of the cluster assignment probability for a given point through $e(\mathbf{x}_i) = -\sum_{k=1}^K q_{ki} \log q_{ki}$. We use this quantity to characterize the quality of our results in subsequent experiments.

9 Experiments and Results

In this section, we describe the performance of the Mixture Density Kernels on a synthetic clustering problem and on a real-world image segmentation problem. For the synthetic data set, we show that the algorithm produces superior results when compared with a standard Gaussian kernel.

9.1 Clustering with Mixture Density Mercer Kernels on Synthetic Data Figure 1 shows a plot of the two-dimensional synthetic data. These data are

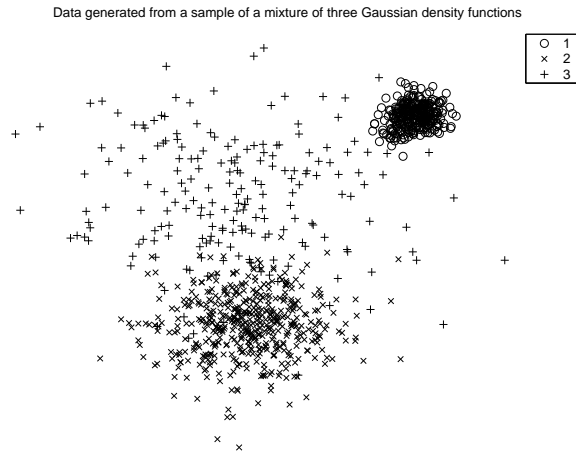


Figure 1: This two dimensional synthetic data was generated by sampling a Gaussian Mixture Density with prior probabilities $P(c) = [0.3, 0.5, 0.2]$, means $\mu_1 = [2, 3.5], \mu_2 = [0, 0], \mu_3 = [0, 2.0]$, and standard deviations $\sigma_1 = 0.2, \sigma_2 = 0.5, \sigma_3 = 1.0$. Clusters 2 and 3 overlap due to their larger standard deviations.

generated from a mixture of three Gaussians. The first Gaussian, labelled with the symbol 'o' has a small variance compared with the other two Gaussians. The second Gaussian has a larger spread, and the third Gaussian partially overlaps the first and second [14]. We generated 10,000 points for training, testing, and evaluation of the model. We recognize that this amount of data is very large to specify the model, but wanted to also obtain an estimate of the performance of the algorithm on a moderate sized data set.

The first experiment consists of computing the Kernel matrix for the synthetic data using a Gaussian Kernel as well as the Mixture Density Mercer Kernels. Since we generated the data, we are able to compute the error between the calculated kernel matrix and the optimal kernel matrix. In the best case, where there is no error in classification, the kernel matrix should be a block diagonal matrix, since each point falls in only one cluster. For the Gaussian kernel shown in Figure 2 we see that Class 1 is almost perfectly classified, i.e., the kernel matrix is nearly square in the upper right hand corner. Class 2 has a higher degree of error as depicted by the larger size squares in the lower, off diagonal elements. Class 3 has the poorest performance, since its class distribution is washed between the other two classes, as exhibited by the lack of a clear block structure. The mean squared error between the ideal block diagonal matrix and the calculated Gaussian

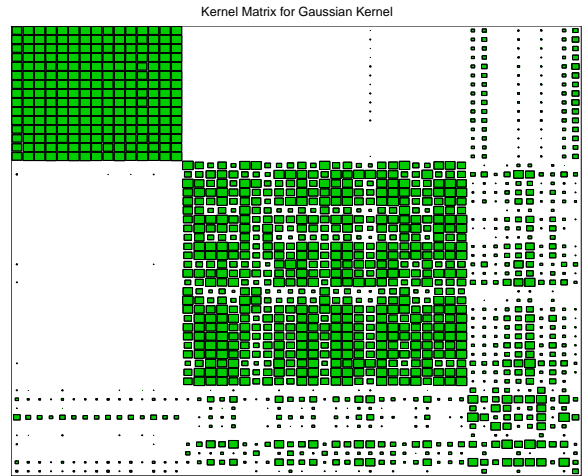


Figure 2: This is a representation of the Gaussian Kernel matrix for the synthetic data show in Figure 1. The size of the square corresponds to the magnitude of the kernel value (maximum value is 1). Class 1 exhibits a very clear clustering, whereas Class 3 shows a high degree of error and confusion with the other classes. The mean squared error between the 'true' kernel matrix and the calculated Gaussian kernel matrix is $0.09 \pm 10\%$.

kernel matrix is approximately $0.09 \pm 10\%$.

Figure 3 shows the kernel matrix for the Mixture Density Mercer Kernels. We generated this matrix by taking $C_m = 5$ for all m and $M = 50$, which represents the number of clusters in the ensemble models, and the number of ensembles, respectively. We initialized the EM algorithm with the initial centers from a k-means algorithm. We assumed spherical covariance matrices but put no further prior information into the model.

This figure illustrates that, as in Figure 2, Cluster 1 is well defined with nearly all the density in the matrix in the upper left hand corner. However, unlike the Gaussian kernel, the Mixture Density kernel better distinguishes Clusters 2 and 3. In fact, Cluster 3 is a well formed block with some errors made in assigning points to Cluster 2. However, these points may be in regions where there is significant overlap between the two Gaussians that generated these points. The mean squared error for this kernel matrix is $0.04 \pm 10\%$. Building the Mixture Density Kernel matrix on the 10,000 points in the training and test sets took approximately 10 seconds on a 3 GHz linux machine with dual Xeon processors.

We performed a one-tailed t-test to determine whether the difference in these matrices is statistically significant. We built 100 Mixture Density Mercer Ker-

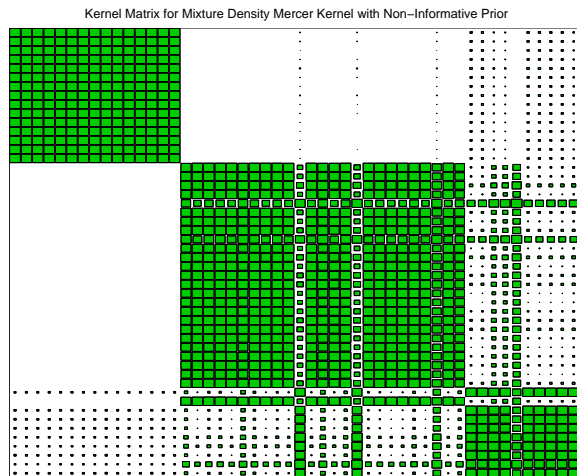


Figure 3: The kernel matrix generated from the Mixture Density Mercer Kernels. While Class 1 continues to be clearly demarcated in the upper left hand region as with the Gaussian Kernel, this kernel also does a better job at distinguishing between Clusters 2 and 3. The mean squared error between the correct kernel matrix and the estimated kernel matrix is $0.04 \pm 10\%$. Notice that some points are not correctly classified by this kernel as indicated by the dark vertical lines in the off-diagonal regions of the matrix. These lines correspond to points in the data space which arise from overlapping modes of the mixture density.

nels and compared the mean squared error between the true kernel matrix and the MDMK matrix. The mean squared errors are in fact statistically significant at the 1% level with $p < 6.6e - 190$, and $N = 100$. While this result is not completely unexpected, it validates the intuition that the Mixture Density Kernel can correctly characterize data that have multiple modes, but can also help reduce the uncertainty in regions where there are overlaps between modes. In this case, the data generating process matched the model used in the sense that both were Gaussian distributed. However, the model has 5 modes, as opposed to the correct value of 3.

Our studies indicate that the choice of the number of modes does not greatly effect the distribution of points in the Kernel matrix as long as the model has sufficient complexity. Figure 4 shows that as the model complexity increases beyond the true number of modes in the data generating process, which is three for this example (see Figure 1), an increase in model complexity by a factor of 6 does not make a significant difference in the estimated kernel. The error bars are computed by taking the standard deviation across 25 runs.

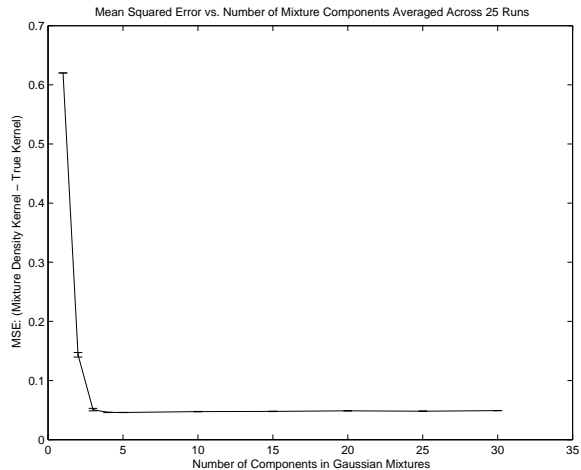


Figure 4: This plot shows the mean squared error with error bars between the estimated kernel matrix using the Mixture Density Mercer Kernel technique and the true kernel matrix, averaged over 25 runs. Notice that once the model has sufficient complexity, large changes in the model parameters have little effect on the estimated kernel.

9.2 Image Segmentation over Snow and Ice

In this section, we describe the performance of the Mixture Density Mercer Kernel algorithm on a real-world image segmentation problem. We begin by giving a brief motivation of the problem, followed by an analysis of the results. The paper by Srivastava and Stroeve [17] contains a discussion of these ideas in the area of onboard remote sensing.

The detection of clouds within a satellite image is essential for retrieving surface geophysical parameters from optical and thermal imagery. Operational surface albedo and temperature products require the cloud-detection because the retrieval methods are valid for clear skies only. Even a small percentage of cloud cover within a radiometer pixel can affect in such a way that determination of surface variables, such as albedo and temperature becomes impossible. Thus, routine processing of satellite data requires reliable automated cloud detection algorithms that are applicable to a wide range of surface types. Unfortunately, cloud-detection, particularly over snow- and ice-covered surfaces is a problem that has plagued working with optical and thermal imagery since the first satellite-imaging sensor. Cloud-detection over snow and ice is difficult due to the lack of spectral contrast between clouds and snow. However, spectral information in the shortwave infrared, texture patterns, and other features may be

used together to detect cloud contamination.

Common approaches to detecting cloud cover are based on spectral contrast, radiance spatial contrast, radiance temporal contrast, or a combination of these methods. These techniques work well over dark targets (e.g. vegetation), since clouds appear brighter (higher albedo) in the visible range, and have lower temperatures in the infrared compared to the cloud-free background. Threshold values are then chosen to represent the cloud-free background. Problems with this method however, are that different thresholds typically need to be selected from scene to scene. Another type of cloud detection that does not require absolute thresholds evaluates the spatial coherence of the observed scene. However, coherence tests suffer because false detection is likely for clear pixels directly adjacent to cloud pixels.

We obtained MODIS level 1B data for the Greenland ice sheet from the NASA Langley DAAC and mapped the data to a 1.25 km equal-area scalable Earth-grid (EASE-grid) using software developed by NSIDC to process MODIS level 1B data and convert the visible channel data to top-of-the-atmosphere (TOA) reflectances. Next the TOA reflectances were normalized by the cosine of the solar zenith angle. Only the first 7 MODIS channels were used for this study. The image shown in Figure 5 was taken on day 188 of the year 2002 and is the output of Channel 6, which is tuned to detect clouds. Figure 6 shows the corresponding test image, which was taken on day 167 of the year 2002. We are unable to show the images in the other 5 bands due to space limitations. As expected, the spectral signals for the 7 different MODIS channels are highly correlated, with linear correlation coefficients over 98%. We used this data as a training set for the Mixture Density Mercer Kernel. We prepared the data for use in the algorithm by taking the first difference across the spectral bands thus yielding 6 differenced bands and then building 5×5 blocks of the data. This procedure results in a 150 dimensional vector representing each pixel, where $150 = 5 \times 5 \times 6$. We have found that the differencing procedure yields improved results for the Mixture Density Kernel as well as for other detection algorithms.

Figure 7 shows the result of learning a standard Gaussian Mixture Model with 10 modes on the training data in Figure 5, and applying that model to the test data shown in Figure 6. The plots are a color coding of the maximum posterior probability class label for the Gaussian Mixture Model. The model does a good job at segmenting the image, and reveals the large three-pronged cloud over the ice sheet. However, notice that the cloud is characterized as a single monolithic entity with little structure depicted within it. A careful comparison of the same region with the test image in

<i>Channel</i>	<i>Spectral (nm)</i>	<i>Wavelength</i>
1	620-670	
2	841-876	
3	459-479	
4	545-565	
5	1230-1250	
6	1628-1652	
7	2105-2155	

Table 1: The bandwidths for the first seven MODIS channels. The spatial resolutions for Channels 1 and 2 are 250 m, and 500 m for Channels 3-7.

Figure 6 shows that there is considerable structure to the cloud that is missed. Furthermore, other regions, such as the large cloud in the lower right hand portion of the image are broken into multiple constituents, whereas Channel 6 does not show such structure.

Figure 8 shows the results of applying the same data to the Mixture Density Kernel. We built the kernel on the same training data, and used 10 mixtures in each model in the ensemble. We built 50 ensembles, resulting in a 500 dimensional feature space. The figure shows a plot of the maximum probability class label, called q_{ki} in Equation 8.5 for each pixel. The first area to notice is the structure of the three-pronged cloud over the ice sheet. This structure indicates that the cloud is not a homogeneous entity, but has several constituents. The western side of the image also shows a considerable amount of variation, particularly in the clouds in the lower part of the image. We point out these details in contrast to the results for the Gaussian Mixture Model for the same region. These results suggest that the Mixture Density Kernel can reveal new features in the data in some cases better than the Gaussian Mixture Model. However, such variation can be found in other regions of the image, thus making an objective evaluation difficult. This difficulty is not unique to Mixture Density Kernels and applies equally to general unsupervised learning algorithms.

10 Discussion and Conclusions

We have shown a method to generate a Mercer Kernel function from an ensemble of mixture models. The Mixture Density Mercer Kernel function is the dot product of the vectors of class distributions across ensembles. We have shown that this function can be interpreted as a new similarity measure between two data points, and has the feature that regions of higher uncertainty can be reduced. We exhibited the

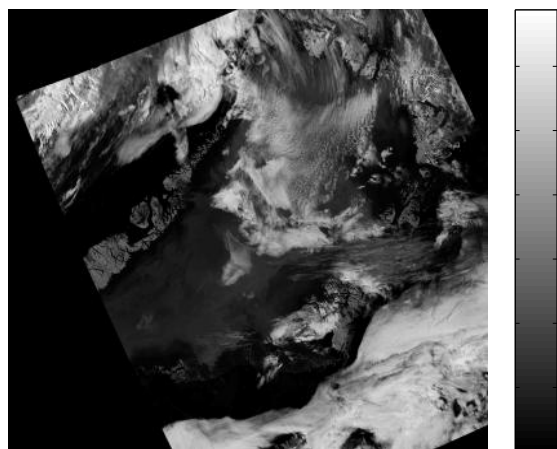


Figure 5: This figure shows the output of Channel 6 for day 188 in the year of 2002 from the MODIS instrument. Clouds are characterized by regions of greater density of white. This data was used for training the Mixture Density Mercer Kernel and the GMM.

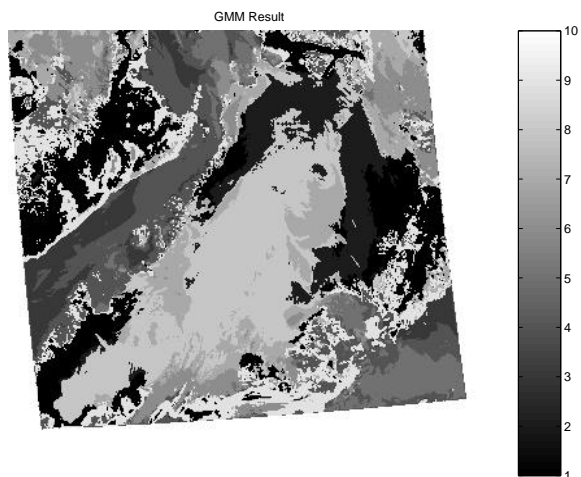


Figure 7: The results of applying a Gaussian Mixture Model to the test data using 10 mixture coefficients. This model does a good job at segmenting the image across the seven spectral bands. Notice that the large three-pronged cloud is characterized as a single entity.

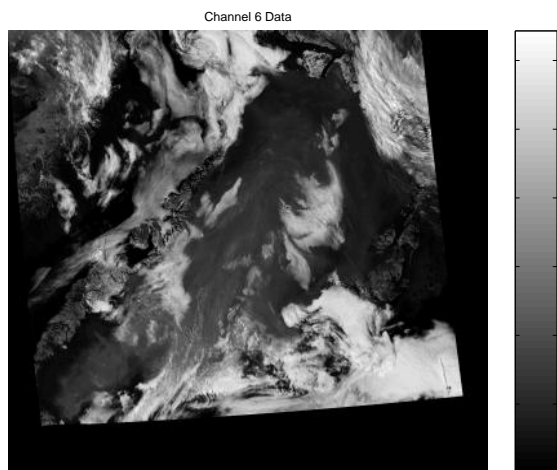


Figure 6: This figure shows the output of Channel 6 for day 167 in the year of 2002 from the MODIS instrument. Clouds are characterized by regions of greater density of white. This data was used to test the Mixture Density Mercer Kernel and the Gaussian Mixture Model. Regions of clouds that are over the ice sheet in the central part of the figure can be harder to detect than clouds that are over water or other regions, because of the lack of spectral contrast. The tree-pronged cloud in the center of the figure is directly over the Greenland ice sheet.

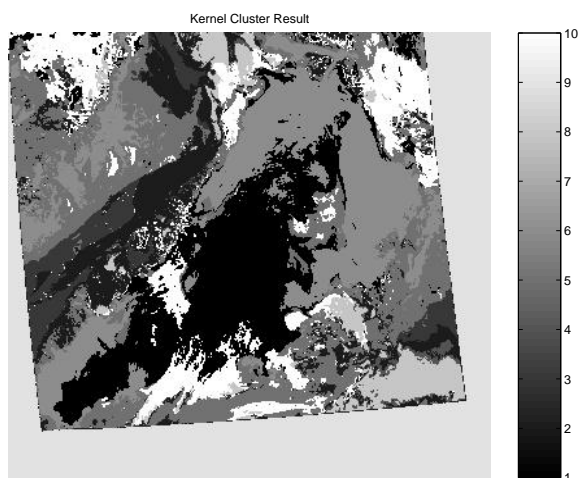


Figure 8: This figure shows the results of the Mixture Density Mercer Kernel on the test data using 10 mixtures in the ensemble with 50 models in total. In comparison to the Gaussian Mixture Model, these results show a difference in the characterization of the three-pronged cloud over the ice sheet. This cloud shows additional structure using two different elements. Other areas of comparison are the three clouds at the top of the image. The Mixture Density Mercer Kernel method correctly segments these into the same class, whereas there is considerable variation in the competing model.

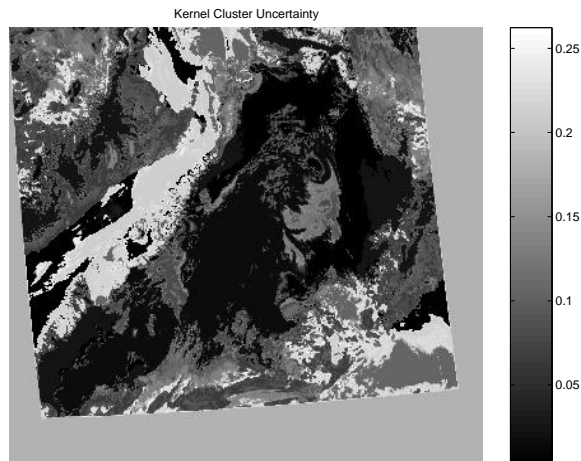


Figure 9: This figure shows the uncertainty in the cluster assignment as shown by the entropy of class distribution. It is interesting to note that there is very low uncertainty over most of the ice sheet, even though it is divided into two entities. The three-pronged cloud over the ice sheet shows up as a region with higher uncertainty, as one would expect.

algorithm in an unsupervised clustering setting, where it shows some promise. We have also shown that the Mixture Density Mercer Kernel can incorporate domain knowledge through its inclusion as a prior distribution on the ensemble models. While the algorithm has some advantages, it is expensive to compute and still exhibits some run-to-run variation. We are currently researching methods to further reduce this variation.

This paper has shown the behavior of the new kernel on unsupervised clustering problems. We are currently preparing the results of using this kernel function for supervised prediction problems, in particular on the snow, ice, and cloud classification.

11 Acknowledgements

The author would like to thank Bill Macready, Nikunj Oza, and Julianne Stroeve and the reviewers for valuable discussions and feedback. Dr. Stroeve also provided the MODIS data used as examples in this paper. This work was supported by the NASA Intelligent Data Understanding segment of the Intelligent Systems Program.

References

[1] Strehl A. and J. Ghosh, *Cluster ensembles a knowledge reuse framework for combining multiple partitions*,

Journal of Machine Learning Research **3** (2002), 583–617.

[2] L. Breiman, *Bagging predictors*, Machine Learning **26** (1996), 123–140.

[3] W. L. Buntine, *Operations for learning with graphical models*, Journal of Artificial Intelligence Research **2** (1994), no. 1, 159–225.

[4] P. Cheeseman and J. Stutz, *Bayesian classification (autoclass): Theory and results*, Advances in Knowledge Discovery and Data Mining (1995).

[5] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, *On kernel target alignment*, Journal of Machine Learning Research (2002).

[6] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines*, Cambridge University Press, 2000.

[7] A. P. Dempster, M. Laird, N., and D. B. Rubin, *Maximum likelihood from incomplete data via the em algorithm*, Journal of the Royal Statistical Society B (1977).

[8] M. Girolami, *Mercer kernel based clustering in feature space*, IEEE Transactions on Neural Networks **13** (2001), no. 4, 780–784.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer, 2001.

[10] D. Haussler, *Convolution kernels on discrete structures*, Tech. report, University of California Santa Cruz, 1999.

[11] T. Jaakkola and D. Haussler, *Exploiting generative models in discriminative classifiers*, Advances in Neural Information Processing Systems **11** (1999).

[12] W. G. Macready, *Density estimation with mercer kernels*, Technical Report TR03.13 of the Research Institute of Advanced Computer Science (2003).

[13] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*, Academic Press, 1979.

[14] I. T. Nabney, *Netlab*, (2001).

[15] D. Ormoneit and V. Tresp, *Improved gaussian mixture density estimates using bayesian penalty terms and network averaging*, Advances in Neural Information Processing Systems, vol. 8, 1995.

[16] P. Smyth and D. Wolpert, *Stacked density estimation*, Advances In Neural Information Processing Systems **10** (1998).

[17] A. N. Srivastava and J. Stroeve, *Onboard detection of snow, ice, clouds, and other geophysical processes using kernel methods*, Proceedings of the ICML 2003 Workshop on Machine Learning Technologies for Autonomous Space Applications, 2003.

[18] K. Tsuda, M. Kawanabe, G. Ratsch, S. Sonnenburg, and K.R. Muller, *A new discriminative kernel from probabilistic models*, Neural Computation (2002).