The mixture of multi-kernel relevance vector machines model

Konstantinos Blekas and Aristidis Likas

Department of Computer Science, University of Ioannina
PO.Box 1186, Ioannina 45110 - Greece
E-mail:{kblekas, arly}@cs.uoi.gr

***Abstract*—We present a new regression mixture model where each mixture component is a multi-kernel version of the Relevance Vector Machine (RVM). In the proposed model, we exploit the enhanced modeling capability of RVMs due to their embedded sparsity enforcing properties. Moreover, robustness is achieved with respect to the kernel parameters, by employing a weighted multi-kernel scheme. The mixture model is trained using the maximum a posteriori (MAP) approach, where the Expectation Maximization (EM) algorithm is applied offering closed form update equations for the model parameters. An incremental learning methodology is also presented to tackle the parameter initialization problem of the EM algorithm. The efficiency of the proposed mixture model is empirically demonstrated on the time series clustering problem using various artificial and real benchmark datasets and by performing comparisons with other regression mixture models.**

***Keywords*-Relevance Vector Machines; mixture models; sparse prior; multi-kernel; incremental EM learning**

I. INTRODUCTION

Mixture model training constitutes a flexible and well established approach in the case of data sets containing data objects that have been generated from heterogeneous sources [1], [2]. Among many advantages they offer, mixture models provide a nice framework for cluster analysis by assigning objects to mixture components (or clusters) while simultaneously estimating the model parameters. *Regression mixture models* are a special type of mixture models where the components correspond to regression functions and they have mainly employed to model *sequential data*.

Many problems of scientific interest can be formulated as sequential data modeling problems. Such type of data can be encountered in a number of diverse applications, ranging from gene clustering in bioinformatics to clustering of cyclone trajectories [3], [4], [5], and recently to video surveillance problems [6], [7], [8] and motion recognition [9].

A natural framework for modeling sequential data is through regression mixture models, also known as latent class regression analysis [1], [2]. A regression mixture model allows for simultaneously modeling heterogeneous regression functions by training a mixture of distinct distributions, each corresponding to a latent class. Objects within each latent class share the same regression function. Through the literature there are different types of regression mixture models that have been used for sequential data modeling [10]. Among them, Hidden Markov Models [11], polynomial and spline regression models [12],[4],[5], mixtures of ARMA models [13] and mixtures of Gaussian processes [14] are commonly used models. These methods are suffering from the drawback of not automatically addressing the problem of model order selection, which is very important in regression. If the order of the regressor model is too large, it overfits the observations and does not generalize well. On the other hand if it is too small, it might miss trends in the data.

Sparse Bayesian regression offers a solution to the model selection problem, see for example [15], [16], [17] and [18], by introducing sparse priors on the model parameters. Enforcing sparsity is a fundamental machine learning regularization principle and has been successfully used to tackle several problems, such as feature selection. The key idea behind sparse Bayesian regression is that we can obtain more flexible inference methods by employing models having initially many degrees of freedom and applying a heavy tail prior is imposed to the coefficients of the regressor. During training, the coefficients that are not significant are zeroed out due to the prior, thus only a few coefficients are retained in the model which are considered significant for the particular training data.

In this paper we propose a regression mixture model where each component corresponds to an extension of the typical RVM model [15] assuming a *weighted multikernel* function, ie. each RVM kernel is a weighted combination of basic kernels and the combination weights are estimated during training [19]. We call this extension Multi-kernel RVM (MKRVM). In the RVM model the marginal distribution of the observations given the hyperparameters is a Gaussian distribution (see Eq. 9). Therefore, the regression mixture is converted into a typical mixture model of Gaussians with zero mean and full covariances. A significant problem in regression is how to define the scalar parameter of the kernel design matrix. In this study we have faced this issue by considering a multi-kernel scheme where we assume that each mixture component has a unique kernel matrix calculated as a linear combination of a (common) set of matrices with known kernel parameter values. These vectors of coefficients are part of the mixture model parameters which must be estimated. Then, a maximum a posteriori expectation maximization algorithm (MAP-EM) [20], [1] is applied to learn this mixture of multi-kernel RVMs model and fit the input data. This is leads to update rules of all model parameters in closed form during the M-step and improves data fitting. In the case of the multi-kernel scheme coefficients this leads to a convex quadratic programming problem with constraints. Another contribution of the present work is an incremental scheme for training the mixture of multi-kernel RVMs model that is based on an appropriate repeated splitting process. This causes to make the learning process independent of the initialization of model parameters and obtain optimum solutions.

The proposed regression mixture models can be employed to model and cluster a *set of functions*, ie. each object in the training set is a function represented as by a set of (input, output) pairs. In this work we have focused on the specific case of *time series clustering*, ie. the (input, output) pairs of a data object correspond to a particular time series. The performance of the proposed methodology is evaluated using a variety of simulated and real data sets. Comparative results demonstrate improvements over previous methods such as the polynomial regression mixture model and the mixture of autoregressive models. Since the ground truth is already known for all datasets, we have used the percentage of correct classification (purity) and the normalized mutual information (NMI) quantities for evaluating the performance of each method. Also, in the case of artificial data, we have computed as a performance metric the error in estimating the original functions that have generated each cluster. As experiments indicate, our method offers greater flexibility and robustness and obtains superior clustering solutions.

In section 2 we describe the multi-kernel relevance vector machine which is the building block in our approach. The proposed sparse regression mixture model is then presented in section 3, along with the EM algorithm used for parameter estimation and the incremental learning procedure. To assess the performance of the proposed methodology we present in section 4 numerical experiments with artificial and real benchmark data sets. Finally, in section 5 we provide conclusions and suggestions for future research.

II. The Multi-Kernel Relevance Vector Machine

Suppose we are given a set of N samples (data objects) $\boldsymbol{Y} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\}$, where each sample $\boldsymbol{y}_n$ consists of L input-target pairs: $\boldsymbol{y}_n = \{\boldsymbol{x}_n, \boldsymbol{t}_n\} = \{(\boldsymbol{x}_{ni}, t_{ni})\}_{i=1}^{L}$.

In regression problems we consider that the real target values t_{ni} correspond to noisy measurements of the output of a parametric model f with input vector $\boldsymbol{x}_{ni}$, i.e.

$$t_{ni} = f(\boldsymbol{x}_{ni}; \theta) + \epsilon_{ni} \ , \tag{1}$$

where ϵ_{ni} refers to noise and θ denotes the model parameters which must be estimated using a training set. Moreover, for the conditional density of each sample $\boldsymbol{y}_n$ we can write

$$p(\boldsymbol{y}_n = \{\boldsymbol{x}_n, \boldsymbol{t}_n\}|\theta) = p(\boldsymbol{t}_n|\boldsymbol{x}_n, \theta)p(\boldsymbol{x}_n) \propto p(\boldsymbol{t}_n|\boldsymbol{x}_n, \theta) \tag{2}$$

Typically, we can model $\boldsymbol{t}_n$ by assuming an M-order linear regression model on the L input vectors with an additive noise term given by

$$\boldsymbol{t}_n = \Phi_n \boldsymbol{w} + \boldsymbol{\epsilon}_n \ , \tag{3}$$

where $\boldsymbol{w} = (w_1, \dots, w_M)^T$ is the vector with the unknown regression coefficients, and Φ_n is the design matrix of size $L \times M$. In the above model, the error term $\boldsymbol{\epsilon}_n$ is a L-dimensional vector that is assumed to be zero mean Gaussian with a spherical covariance, i.e. $\boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \sigma^2 I)$.

For constructing the design matrix Φ we can employ several approaches. A simple approach is to use the Vandermonde or B-splines matrix, in cases where we assume polynomial or splines regression models, respectively [21]. Another option is to consider a *kernel* design matrix of size $L \times L$, consisting of L basis functions, $\Phi_n = [\phi(\boldsymbol{x}_{n1}), \dots, \phi(\boldsymbol{x}_{nL})]$ where $\phi(\boldsymbol{x}_{ni})$ is a vector of L kernel values among $\boldsymbol{x}_{ni}$ and all other inputs, i.e. $\phi(\boldsymbol{x}_{ni}) = (K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{n1}), \dots, K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{nL}))$. This is achieved by appropriately selecting a kernel function, with the RBF kernel function to be the most commonly used:

$$K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{nk}) = \exp(-\frac{\|\boldsymbol{x}_{ni} - \boldsymbol{x}_{ik}\|^2}{2\lambda}) \ .$$

The scalar parameter λ plays a significant role to the quality of the fitting procedure. Its selection depends on the amount of local variations of input data sequences.

In our case we consider a multi-kernel scheme by using a discrete set of S RBF kernel functions K_s, each one having its own scalar parameter value λ_s. In particular, we assume that the composite kernel matrix Φ_n can be written as a linear combination of S kernel matrices F_{ns}:

$$\Phi_n = \sum_{s=1}^{S} u_s F_{ns} \ , \tag{4}$$

where the coefficients u_s satisfy the constraints $u_s \geq 0$ and $\sum_{s=1}^{S} u_s = 1$. These parameters should be estimated during learning in order to construct the composite kernel matrix, as it will be shown later. It must be noted that the multi-kernel idea has been successfully used in several machine learning models [22], [23], [24], [19], that assume a weighted linear sum of kernel and estimate the kernel weights during training. However, to the best of our knowledge this is the first time that a multi-kernel version of RVM with adaptive kernel weights is proposed.

Using Equation 3, it is obvious that given the set of regression parameters $\{\boldsymbol{w}, \sigma^2, \boldsymbol{u}\}$, we can model the conditional probability density of the target $\boldsymbol{t}_n$ with the normal distribution, i.e.

$$p(\boldsymbol{t}_n|\boldsymbol{w}, \sigma^2, \boldsymbol{u}) = \mathcal{N}(\boldsymbol{t}_n|\Phi_n \boldsymbol{w}, \sigma^2 I) \ . \tag{5}$$

An important issue, when using a regression model is how to define its order M, since models of small order may lead to underfitting, while large values of M may lead to overfitting. One approach to tackle this problem is the Bayesian regularization method that has been successfully employed in the Relevance Vector Machine (RVM) model [15]. This technique initially assumes a large value of the order M ($M = L$) and imposes a heavy tailed prior distribution $p(\boldsymbol{w})$ on the regression model parameters w_i, to zero out most of them after training.

More specifically, the prior is defined in a hierarchical way by considering a zero-mean Gaussian distribution over $\boldsymbol{w} = (w_1, \ldots, w_L)$:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, A^{-1}) = \prod_{i=1}^{T} \mathcal{N}(w_i|0, \alpha_i^{-1}) \qquad (6)$$

where A is a diagonal matrix containing L elements of the hyperparameter vector $\boldsymbol{\alpha} = [\alpha_1 \ldots \alpha_L]$. In addition, a Gamma prior is imposed in each hyperparameter α_i:

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{T} Gamma(\alpha_i|a, b) \propto \prod_{i=1}^{T} \alpha_i^{a-1} e^{-b\alpha_i} \ . \qquad (7)$$

Also we can assume a Gamma hyperprior over the noise parameter σ^2:

$$p(\sigma^{-2}) = Gamma(\sigma^{-2}|c, d) \propto \sigma^{-2(c-1)} e^{-d\sigma^{-2}} \ . \qquad (8)$$

All Gamma parameters $\{a, b, c, d\}$ are a priori set to zero values to achieve uninformative priors.

The above two-stage hierarchical prior on α_i is actually a *Student-t* distribution and is called *sparse* [15], since it enforces most of the values α_i to be large, thus the corresponding coefficients w_i are forced to zero and eliminated from the model. In this way the complexity of the regression model is controlled in an automatic and elegant way and overfitting is avoided. By integrating out the contribution of weights $\boldsymbol{w}$ from Equation 5, we can obtain the marginal distribution of target $\boldsymbol{t}_n$ given the model parameters $\theta = \{\boldsymbol{a}, \sigma^2, \boldsymbol{u}\}$, as a zero mean Normal distribution:

$$p(\boldsymbol{t}_n|\theta) = \int p(\boldsymbol{t}_n|\boldsymbol{w}, \sigma^2, \boldsymbol{u}) p(\boldsymbol{w}|\boldsymbol{\alpha}) d\boldsymbol{w} = \mathcal{N}(0, C_n) \ , \qquad (9)$$

where the covariance matrix has the form:

$$C_n = \Phi_n A^{-1} \Phi_n^T + \sigma^2 I \ . \qquad (10)$$

Furthermore, we can obtain the posterior distribution over the weights $\boldsymbol{w}$, which is also Gaussian, as:

$$p(\boldsymbol{w}|\boldsymbol{t}_n, \theta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_n, \Sigma_n) \ , \qquad (11)$$

with mean and covariance given by

$$\boldsymbol{\mu}_n = \sigma^{-2} \Sigma_n \Phi_n^T \boldsymbol{t}_n \ , \ \ \Sigma_n = (\sigma^{-2} \Phi_n^T \Phi_n + A)^{-1} \ . \qquad (12)$$

Thus, the $\Phi_n \boldsymbol{\mu}_n$ denotes the final model-based estimation for sample $\boldsymbol{y}_n$.

Learning the linear weights $\boldsymbol{u}$ of the multi-kernel scheme can be done using the fact that

$$\Phi_n \boldsymbol{\mu}_n = \sum_{s=1}^{S} u_s F_{ns} \boldsymbol{\mu}_n = \mathcal{F}_n \boldsymbol{u} \ , \text{ where}$$

$\mathcal{F}_n = [F_{n1}\boldsymbol{\mu}_n \ \ F_{n2}\boldsymbol{\mu}_n \ \ \cdots \ \ F_{nS}\boldsymbol{\mu}_n]$, and by solving the following minimization problem:

$$\min_{\boldsymbol{u}} \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{t}_n - \mathcal{F}_n \boldsymbol{u}\|^2 \text{ s.t. } \sum_{s=1}^{S} u_s = 1 \text{ and } u_s \geq 0 \ . \qquad (13)$$

More comprehensively, we can rewrite the above objective function as follows:

$$\min_{\boldsymbol{u}} \left\{ \frac{1}{2} \boldsymbol{u}^T \mathcal{Z} \boldsymbol{u} + \boldsymbol{u}^T \boldsymbol{q} \right\}, \text{ s.t. } \sum_{s=1}^{S} u_s = 1 \ , u_s \geq 0 \ , \qquad (14)$$

where $\mathcal{Z} = \sum_{n=1}^{N} \mathcal{F}_n^T \mathcal{F}_n$ and $\boldsymbol{q} = -\sum_{n=1}^{N} \mathcal{F}_n^T \boldsymbol{t}_n$. This is a typical convex quadratic programming problem with both equality and inequality constraints that can be solved by *active-set* methods that use Lagrange multipliers leading to closed form analytical expressions [25].

III. The mixture of MKRVMs model

In the mixture of MKRVMs model there are K multi-kernel RVM components with parameters $\theta_j = \{\boldsymbol{\alpha}_j, \sigma_j^2, \boldsymbol{u}_j\}$, $j = 1, \ldots, K$. According to Eq. 9, each component defines a zero mean Normal distribution:

$$p(\boldsymbol{t}_n|\theta_j) = \mathcal{N}(\boldsymbol{t}_n|0, C_{nj}) \ , \qquad (15)$$

with

$$C_{nj} = \Phi_{nj} A_j^{-1} \Phi_{nj}^T + \sigma_j^2 I \text{ and } \Phi_{nj} = \sum_{s=1}^{S} u_{js} F_{ns} \ . \qquad (16)$$

Moreover, A_j in Eq. 16 is a diagonal matrix containing the elements of hyperparameter vector $\boldsymbol{\alpha}_j$, i.e. $A_j = diag\{\alpha_{j1}, \ldots, \alpha_{jL}\}$. The MK-RVM mixture model is described by the following probability density function:

$$f(\boldsymbol{t}_n|\Theta) = \sum_{j=1}^{K} \pi_j p(\boldsymbol{t}_n|\theta_j) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{t}_n|\mathbf{0}, C_{nj}) \ . \qquad (17)$$

Let Θ denote the set of all mixture model parameters, i.e. $\Theta = \{\pi_j, \theta_j\}_{j=1}^{K}$. The mixing weights π_j satisfy the constraints: $\sum_{j=1}^{K} \pi_j = 1$ and $\pi_j \geq 0$. The same happens with the coefficients $\boldsymbol{u}_j$ of the multi-kernel scheme, i.e. $\sum_{s=1}^{S} u_{js} = 1$ and $u_{js} \geq 0$. The parameters $\{\boldsymbol{\alpha}_j, \sigma_j^2\}$ are constrained by Gamma prior distributions:

$$p(\boldsymbol{\alpha}_j) = \prod_{i=1}^{T} Gamma(\alpha_{ji}|a_j, b_j) \propto \prod_{i=1}^{T} \alpha_{ji}^{a_j-1} e^{-b_j \alpha_{ji}} \ , \qquad (18)$$

$$p(\sigma_j^{-2}) = Gamma(\sigma_j^{-2}|c_j, d_j) \propto \sigma_j^{-2(c_j-1)} e^{-d_j \sigma_j^{-2}} \ . \qquad (19)$$

where all Gamma parameters $\{a_j, b_j, c_j, d_j\}$, are set to zero (uninformative priors).

To train the mixture of MKRVMs model we define the maximum a posteriori (MAP) log-likelihood function:

$$L_{MAP} = \log p(\boldsymbol{Y}|\Theta) + \log p(\Theta) = \sum_{n=1}^{N} \log\{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj})\} + \sum_{j=1}^{K}\{\log p(\boldsymbol{\alpha}_j) + \log p(\sigma_j^{-2})\} \ , \tag{20}$$

and use the Expectation-Maximization (EM) algorithm [20] for likelihood maximization. EM performs iteratively two steps: The E-step, where the current posterior probabilities are calculated of any sample $\boldsymbol{y}_n = \{\boldsymbol{x}_n, \boldsymbol{t}_n\}$ to belong to any cluster j:

$$z_{nj} = P(j|\boldsymbol{y}_n, \Theta) = \frac{\pi_j \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj})}{\sum_{j'=1}^{K} \pi_{j'} \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj'})} \ . \tag{21}$$

During the M-step the maximization of the expected value of the complete log-likelihood (Q-function) is performed with respect to Θ. In our case the Q-function is:

$$Q(\Theta) = \sum_{n=1}^{N}\sum_{j=1}^{K} z_{nj}\{\log \pi_j - \frac{1}{2}\log|C_{nj}| - \frac{1}{2}\boldsymbol{t}_n^T (C_{nj})^{-1}\boldsymbol{t}_n\} + \sum_{j=1}^{K}\{\sum_{i=1}^{L}\{a_j \log(\alpha_{ji}) - b_j\alpha_{ji}\} + c_j \log(\sigma_j^{-2}) - d_j\sigma_j^{-2}\} \ , \tag{22}$$

where the quantities z_{nj} have been computed by Eq. 21.

Setting the partial derivatives equal to zeros, the following update rules for the mixture parameters are obtained:

$$\hat{\pi}_j = \frac{\sum_{n=1}^{N} z_{nj}}{N} \ , \tag{23}$$

$$\hat{\alpha}_{ji} = \frac{\sum_{n=1}^{N} z_{nj} + 2a_j}{\sum_{n=1}^{N} z_{nj}\mu_{nji}^2 + \sum_{n=1}^{N} z_{nj}(\Sigma_{nj})_{ii} + 2b_j} \ , \tag{24}$$

$$\hat{\sigma}_j^2 = \frac{\sum_{n=1}^{N} z_{nj}\|\boldsymbol{t}_n - \Phi_{nj}\boldsymbol{\mu}_{nj}\|^2 + 2d_j}{\sum_{n=1}^{N} z_{nj}(L - \sum_{i=1}^{T}\gamma_{nji}) + 2c_j} \ . \tag{25}$$

In the above rules we have used the following expressions [15]:

$$\log|\Phi_{nj}A_j^{-1}\Phi_{nj}^T + \sigma_j^2 I| = -\log|\Sigma_{nj}| + \log\sigma_j^2 - \log|A_j|, \tag{26}$$

$$\boldsymbol{t}_n^T(\Phi_{nj}A_j^{-1}\Phi_{nj}^T + \sigma_j^2 I)^{-1}\boldsymbol{t}_n = \frac{1}{\sigma_j^2}\boldsymbol{t}_n^T(\boldsymbol{t}_n - \Phi_{nj}\boldsymbol{\mu}_{nj}) = \frac{1}{\sigma_j^2}\|\boldsymbol{t}_n - \Phi_{nj}\boldsymbol{\mu}_{nj}\|^2 + \boldsymbol{\mu}_{nj}^T A_j \boldsymbol{\mu}_{nj} \ , \tag{27}$$

where

$$\boldsymbol{\mu}_{nj} = \sigma_j^{-2}\Sigma_{nj}\Phi_{nj}^T\boldsymbol{t}_n \ , \tag{28}$$

$$\Sigma_{nj} = (\sigma_j^{-2}\Phi_{nj}^T\Phi_{nj} + A_j)^{-1} \ . \tag{29}$$

Note also that in the above equations (Eqs. 24, 25) the $(\Sigma_{nj})_{ii}$ indicates the i-th diagonal element of the j-th RVM posterior weight covariance matrix Σ_{nj}, while μ_{nji} is the i-th element of the posterior mean vector $\boldsymbol{\mu}_{nj}$. Also, the quantities $\{\gamma_{nji}\}$ are defined as $\gamma_{nji} = 1 - \alpha_{ji}(\Sigma_{nj})_{ii}$.

As in the case of a single multi-kernel RVM model (see Eq. 13), the linear weights $\boldsymbol{u}_j = (u_{j1}, \ldots, u_{jS})$ of the multi-kernel scheme can be estimated by solving the following minimization problem per component j:

$$\min_{\boldsymbol{u}_j} \frac{1}{2}\sum_{n=1}^{N} z_{nj}\|\boldsymbol{t}_n - \Phi_{nj}\boldsymbol{\mu}_{nj}\|^2 = \min_{\boldsymbol{u}_j} \frac{1}{2}\sum_{n=1}^{N} z_{nj}\|\boldsymbol{t}_n - \sum_{s=1}^{S} u_{js}F_{ns}\boldsymbol{\mu}_{nj}\|^2 = \min_{\boldsymbol{u}_j} \frac{1}{2}\sum_{n=1}^{N} z_{nj}\|\boldsymbol{t}_n - \mathcal{F}_{nj}\boldsymbol{u}_j\|^2 \quad \text{s.t.} \sum_{s=1}^{S} u_{js} = 1 \text{ and } u_{js} \geq 0 \ , \tag{30}$$

where we have considered only the part of the Q-function (Eq. 22) that involves $\boldsymbol{u}_j$. It must be noted here that we assume the posterior mean vector $\boldsymbol{\mu}_{nj}$ and the covariance matrix Σ_{nj} as constants. Also, the matrix $\mathcal{F}_{nj}$ in the above rule has S columns calculated by $F_{js}\boldsymbol{\mu}_{nj}$, i.e. $\mathcal{F}_{nj} = [F_{n1}\boldsymbol{\mu}_{nj} \ \ F_{n2}\boldsymbol{\mu}_{nj} \ \cdots \ F_{nS}\boldsymbol{\mu}_{nj}]$. We can further write the minimization problem in a more convenient way (similar to Eq. 14):

$$\min_{\boldsymbol{u}_j}\left\{\frac{1}{2}\boldsymbol{u}_j^T\mathcal{Z}_j\boldsymbol{u}_j + \boldsymbol{u}_j^T\boldsymbol{q}_j\right\}, \text{ s.t. } \sum_{s=1}^{S} u_{js} = 1 \ , u_{js} \geq 0 \ , \tag{31}$$

where $\mathcal{Z}_j = \sum_{n=1}^{N} z_{nj}\mathcal{F}_{nj}^T\mathcal{F}_{nj}$ and $\boldsymbol{q}_j = -\sum_{n=1}^{N} z_{nj}\mathcal{F}_{nj}^T\boldsymbol{t}_n$. This is a typical convex quadratic programming problem with both equality and inequality constraints [25].

After EM convergence, the assignment of each sample $\boldsymbol{y}_n$ to the K MKRVM components can be made using the maximum value of the posterior probabilities z_{nj} (Eq. 21). The MKRVM function can be also obtained for each component j as follows:

$$\boldsymbol{w}_j = (\sigma_j^{-2}\sum_{n=1}^{N} z_{nj}\Phi_{nj}^T\Phi_{nj} + A_j)^{-1}\sigma_j^{-2}\sum_{i=1}^{N} z_{nj}\Phi_{nj}^T\boldsymbol{t}_n \ . \tag{32}$$

A. Incremental mixture learning

An important concern when applying the EM algorithm, is its strong dependence on the initialization of the model parameters. Improper initialization may lead to poor local maxima of the log-likelihood, a fact that in turn may affect the quality of the method's estimation capability. A natural way for initialization is to first make a random sampling through the training set $\boldsymbol{Y}$ to select K samples, one for each component. Then, a single multi-kernel RVM is trained using the corresponding selected sample in order to estimate the regression parameters $\theta_j = \{\boldsymbol{\alpha}_j, \sigma_j^2, \boldsymbol{u}_j\}$ for each component j. The mixing parameters π_j are initially set to $\frac{1}{K}$. Finally, one iteration of the EM algorithm (one-step-EM) is executed to further refine these parameters and to evaluate the MAP log-likelihood function value L_{MAP} (Eq. 20). Several such random trials (100 in our experiments) are executed and the solution with the maximum log-likelihood value is selected for initializing the model parameters.

In Gaussian mixture modeling, several methods have been proposed to overcome the problem of poor initialization, which are mainly based on incremental construction of the mixture model[26], [27], [28]. We have adopted such a scheme to train our RVM mixture model and have developed a learning methodology that sequentially adds a new RVM component to the mixture based on a *component splitting* procedure. Initially, we start with a mixture model with one MKRVM component. This is done by executing the single multi-kernel RVM learning process to estimate the initial regression parameters $\{\boldsymbol{\alpha}_1, \sigma_1^2, \boldsymbol{u}_1\}$ following the updated rules given in previous section, where we put $j = 1$ and $z_{nj} = 1$.

Let now assume that we have already computed a mixture f_k with k MKRVM components ($k < K$):

$$f_k(\boldsymbol{t}_n|\Theta_k) = \sum_{j=1}^{k} \pi_j p(\boldsymbol{t}_n|\theta_j) \ . \tag{33}$$

Also, we denote as:

$$f_k^{-j}(\boldsymbol{t}_n|\Theta_k^{-j}) = f_k(\boldsymbol{t}_n|\Theta_k) - \pi_j p(\boldsymbol{t}_n|\theta_j)$$

the model containing the $k-1$ components of the k-order mixture model f_k, after eliminating the contribution of the j-th component.

In order to add a new component, at first an existing component j^* is selected for splitting based on the current maximum mixing prior probability value, i.e. $j^* = \arg\max_j\{\pi_j\}$. A new regression component is then added, labeled $(k+1)$, with weight π_{k+1} ($\pi_{k+1} < \pi_{j^*}$). Thus, the new mixture density function f_{k+1} with $k+1$ components is now given as:

$$\begin{aligned} f_{k+1}(\boldsymbol{t}_n|\Theta_{k+1}) &= f_k^{-j^*}(\boldsymbol{t}_n|\Theta_k^{-j^*}) + \\ \pi_{j^*}{}^{new} p(\boldsymbol{t}_n|\theta_{j^*}) &+ \pi_{k+1} p(\boldsymbol{t}_n|\theta_{k+1}) \ . \end{aligned} \tag{34}$$

The mixing weights of both the new inserted $(k+1)$ and the splitted (j^*) component are initialized as $\pi_{k+1} = \pi_{j^*}{}^{new} = \frac{\pi_{j^*}{}^{old}}{2}$. For initializing the RVM parameters $\theta_{k+1} = \{\boldsymbol{\alpha}_{k+1}, \sigma_{k+1}^2, \boldsymbol{u}_{k+1}\}$ we apply the following strategy: First, we find the samples $\boldsymbol{y}_n$ that currently belong to the cluster j^*. We then select a small percentage of those samples that have the lowest probability (outliers), after sorting them in terms of their density values $p(\boldsymbol{t}_n|\theta_{j^*}) = \mathcal{N}(\boldsymbol{t}_n|0, C_{j^*})$. Next we execute the training procedure of a single multi-kernel RVM component to this this set of samples, in order to obtain an initial estimation of the MKRVM parameters $\theta_{k+1} = \{\boldsymbol{\alpha}_{k+1}, \sigma_{k+1}^2, \boldsymbol{u}_{k+1}\}$. After the above initialization, the EM algorithm is used to estimate the parameters Θ_{k+1} of the new mixture model f_{k+1} with $k+1$ RVM components.

The splitting procedure proceeds in this incremental fashion, adding one MKRVM component at a time, until we receive a mixture model with the desired number (K) of the MKRVM components. This approach is summarized in Algorithm 1. An obvious advantage of the incremental learning scheme is that of simultaneously offering solutions for the intermediate models with $k = 1, \ldots, K$ components. Moreover, it can be exploited in order to introduce criteria to stop the evolution of learning: stop training when the insertion of a new component does not offer any significant improvement of the likelihood function.

Algorithm 1 Incremental learning of the mixture of MKRVMs model

```
   Initially apply the single multiple-kernel RVM training procedure to the dataset $Y$ for estimating parameters $\theta_1 = \{\boldsymbol{a}_1, \sigma_1^2, \boldsymbol{u}_1\}$. Set $\Theta_1 = \{\pi_1, \theta_1\}$ with $\pi_1 = 1$.
1: while $k < K$ do
2:     Select a component for splitting: $j^* = \arg\max_{j=1}^{k}\{\pi_j\}$.
3:     Find samples that currently belong to component $j^*$, i.e. $Y_* = \{\boldsymbol{y}_n : j^* = \arg\max_{j=1}^{k} z_{nj}\}$. Sort them in terms of their density values $p(\boldsymbol{t}_n|\theta_{j^*})$.
4:     Select a subset (e.g. $10\%$) $Y_*^{out}$ of $Y_*$ with the less probable samples (outliers).
5:     Run single multiple-kernel RVM training over $Y_*^{out}$ for initializing new component parameters $\theta_{k+1} = \{\boldsymbol{a}_{k+1}, \sigma_{k+1}^2, \boldsymbol{u}_{k+1}\}$.
6:     Initialize mixing weights as $\pi_{k+1} = \pi_{j^*}{}^{new} = \frac{\pi_{j^*}{}^{old}}{2}$.
7:     Apply the EM algorithm to the new mixture of MKRVMs $f_{k+1}(\boldsymbol{t}_n|\Theta_{k+1})$ and obtain $\Theta_{k+1}$.
8:     $k = k + 1$.
9: end while
```

IV. Experimental Results

We have selected the task of times-series clustering for evaluating the proposed mixture model using a variety of artificial datasets and real benchmarks. In this case, each sample $\boldsymbol{y}_n$ is a sequence of real observations measured at L successive time instances that correspond to the target values t_{ni}. At each time instance the input $\boldsymbol{x}_{ni}$ is a d-dimensional vector that describes the d previous target values, i.e. $\boldsymbol{x}_{ni} = (t_{n,i-d}, t_{n,i-d+1}, \dots, t_{n,i-1})$. During all experiments we have considered inputs of length $d = 10$. Moreover, for constructing the multi-kernel scheme for a dataset, we calculated first the total variance of samples, λ. Next, we used a set of $S = 10$ RBF kernel functions, where each one had a scalar parameter $\lambda_s = k_s\lambda$, where $k_s = [0.1, 0.2, \dots, 1.0]$ (level of percentage). Finally, the linear weights of the multi-kernel scheme were in all cases initialized equally to $u_{js} = 1/S$.

In our study we have tested both the incremental and the typical (with random initialization) regression mixture with MKRVM components, that will be referred next as iMMKRVM and MMKRVM, respectively. In the incremental approach, the hyperparameters α are always initialized as $\alpha_l = 1/L$ (step 5 of Algorithm 1) at every single MKRVM training. We compared our method with two common regression mixture models:

- The polynomial regression mixture model (*MPRM*) that considers a polynomial regression function of order p for any cluster, i.e.

$$t_{ni} = \sum_{l=0}^{p} w_{jl} x_{ni}^{l} \ , \tag{35}$$

  where w_{jl} are the $p+1$ regression coefficients for each cluster. In this case the time instances are considered as inputs ($x_{ni} = i$) and a (common) Vandermonde design matrix is used. Finally, in all experiments we have chosen polynomials of order $p = 10$, since they showed better performance.
- The mixture of autoregressive (*MAR*) model that consists of K different AR models, which correspond to the K clusters of interest. Given a time-series $\boldsymbol{t}_n$ and an order p, the AR(p) model assumes that any value t_{ni} has been generated as a linear combination of p previous values plus a constant term, i.e.

$$t_{ni} = w_{j0} + \sum_{l=1}^{p} w_{jl} t_{n,i-l} \ . \tag{36}$$

  Again, $\{w_{jl}\}_{l=0}^{p}$ are the $p+1$ coefficients for the j-th cluster. In this case, the design matrix is created by setting ones (1) to the first column, while the rest columns have the past p values for every time instance. Experiments have made with setting $p = 10$.

Both regression mixture models were trained using the EM-based maximum likelihood framework [13], [12], [5], where we follow the typical sample-based initialization strategy described in section III-A.

To quantify the performance and measure the quality of the clustering results obtained by each method, we have used two evaluation criteria:

- *purity*, which is the percentage of correctly classified samples after labeling each cluster with the label of the class which is most frequent among the sequences that belong to this cluster, and
- *normalized mutual information* (NMI), which is an information-theoretic measure based on the mutual information of the true labeling (Ω) and the clustering ($\mathcal{C}$) normalized by their respective entropies:

$$NMI(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2} \ , \tag{37}$$

  where

$$I(\Omega, \mathcal{C}) = \sum_k \sum_j P(\omega_k, c_j) \log \frac{P(\omega_k, c_j)}{P(\omega_k) P(c_j)} \tag{38}$$

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) \tag{39}$$

$$H(\mathcal{C}) = -\sum_k P(c_k) \log P(c_k) \ . \tag{40}$$

  The quantities $P(\omega_k)$, $P(c_j)$ and $P(\omega_k, c_j)$ are the probabilities of a sample belonging to class ω_k, cluster c_j and in their intersection, respectively, and are computed based on the corresponding set of cardinalities (frequencies).

A. Experiments with simulated datasets

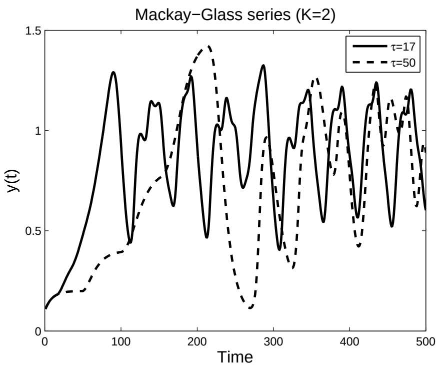
At first, the performance of our method was evaluated using simulated time-series with known ground truth. For this purpose we have selected the Mackey-Glass delay differential equation, which provides a classical benchmark for time-series modeling given by the following rule:

$$r(t+1) = r(t) + \delta \left(0.2 \frac{r(t - \tau/\delta)}{1 + r(t - \tau/\delta)^{10}} - 0.1 r(t) \right) \ , \tag{41}$$

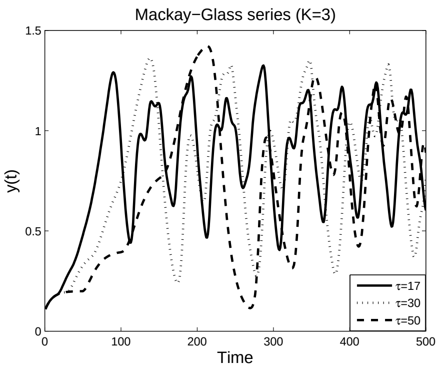
where the step size δ set to $\delta = 0.1$. In our study we have generated three data sets using $K = \{2, 3, 4\}$ of such series of length $L = 500$ respectively, by considering different values for the delay time τ, as illustrated in Fig. 1. A number of 100 noisy copies of the original curve per class were generated using various levels of noise.

Since we are aware of the generative series of each cluster, we have considered an additional evaluation criterion for quantifying the ability of the proposed methodology to model the temporal patterns of the data. In particular, we have computed the mean square error (MSE), between the original Mackey-Glass series $\{\boldsymbol{r}_j, \ j = 1, \dots, K\}$ (Eq. 41), and the estimated mean curves $\bar{\boldsymbol{t}}_j$ after convergence calculated as:

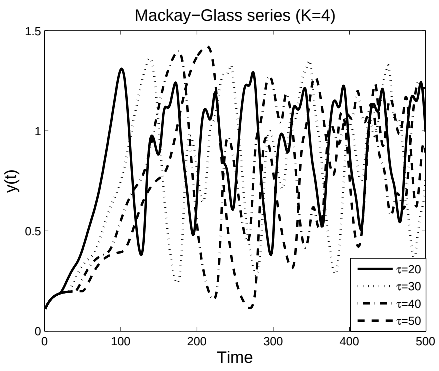
$$\bar{\boldsymbol{t}}_j = \frac{\sum_{n=1}^{N} z_{nj} \Phi_{nj} \boldsymbol{\mu}_{nj}}{\sum_{n=1}^{N} z_{nj}} \ . \tag{42}$$



(a)



(b)



(c)

Figure 1. Three sets with (a) $K = 2$, (b) $K = 3$ and (c) $K = 4$ Mackey-Glass series used for generating artificial datasets.

and

$$MSE = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{L} \|\boldsymbol{r}_j - \bar{\boldsymbol{t}}_j\|^2 \ . \tag{43}$$

For every noise level (SNR) we generated 30 different datasets and we calculated the mean value and the standard deviation of three performance criteria purity, NMI and MSE. Figure 2 illustrates the comparative results in terms of the SNR values. As it is obvious, the proposed mixture of MKRMVs model improves significantly clustering quality as compared to the polynomial and the AR regression mixture,

dataset	# classes (K)	size (N)	dimension (T)
CBF	3	930	128
Coffee	2	56	286
ECG	2	200	96
Face Four	4	112	350
Gun Point	2	200	150
Synthetic control	6	600	60
Trace	4	200	275
Wafer	2	1000	152

Table I
DESCRIPTION OF THE EIGHT (8) UCR DATASETS USED IN OUR EXPERIMENTAL STUDY.

especially for high noise. Between the two proposed versions of MKRVM mixtures, the one based on incremental learning (iMMKRVM) gave slightly better results, confirming its ability to offer efficient parameter initialization and reaching high quality solutions. In what concerns MSE, it is interesting to observe the significant improvement of the fit error criterion in the case of mixture of MKRVMs. This is in agreement with our belief, that sparseness is beneficial both not only for classification accuracy, but also for fitting quality. The MSE results also indicate that the incremental learning approach is superior to the randomly initialized MMKRVM.

B. *Experiments with real benchmarks*

Further experiments have been conducted using various real datasets, obtained from the UCR time series data collection [29], where the ground truth is known. In Table I we present a summary of eight (8) UCR datasets we have used in our study. The results using two evaluation metrics, purity and NMI, are shown in Table II for the two versions of the proposed mixture of MKRVMs and the other two regression mixture models. Since the proposed incremental learning approach (iMMRVM) does not depend on the initialization, we show only the result of a single run. For the rest three methods (MMRVM, MPRM, MAR) we provide the mean value and the standard deviation of each measure (for 30 trials). As it can be observed, the performance of the proposed mixture of MKRVMs is obviously superior and in many cases the difference is quite noticeable.

Finally, Fig. 3 illustrates the mean mixture regression functions as estimated by the proposed iMMKRVM model (according to Eq. 42) in the case of six UCR datasets. From these results a significant conclusion can be drawn, about the impact of the multi-kernel scheme to the regression modeling performance which is affected, sometimes significantly, on the choice of the proper design matrix. In particular, when the input samples contain strong local variations (such as in Coffee and Face Four datasets), the estimated regression should capture these local details using small values of the scalar parameters λ_s. On the contrary, in cases where data samples are smoother (such as in CBF and

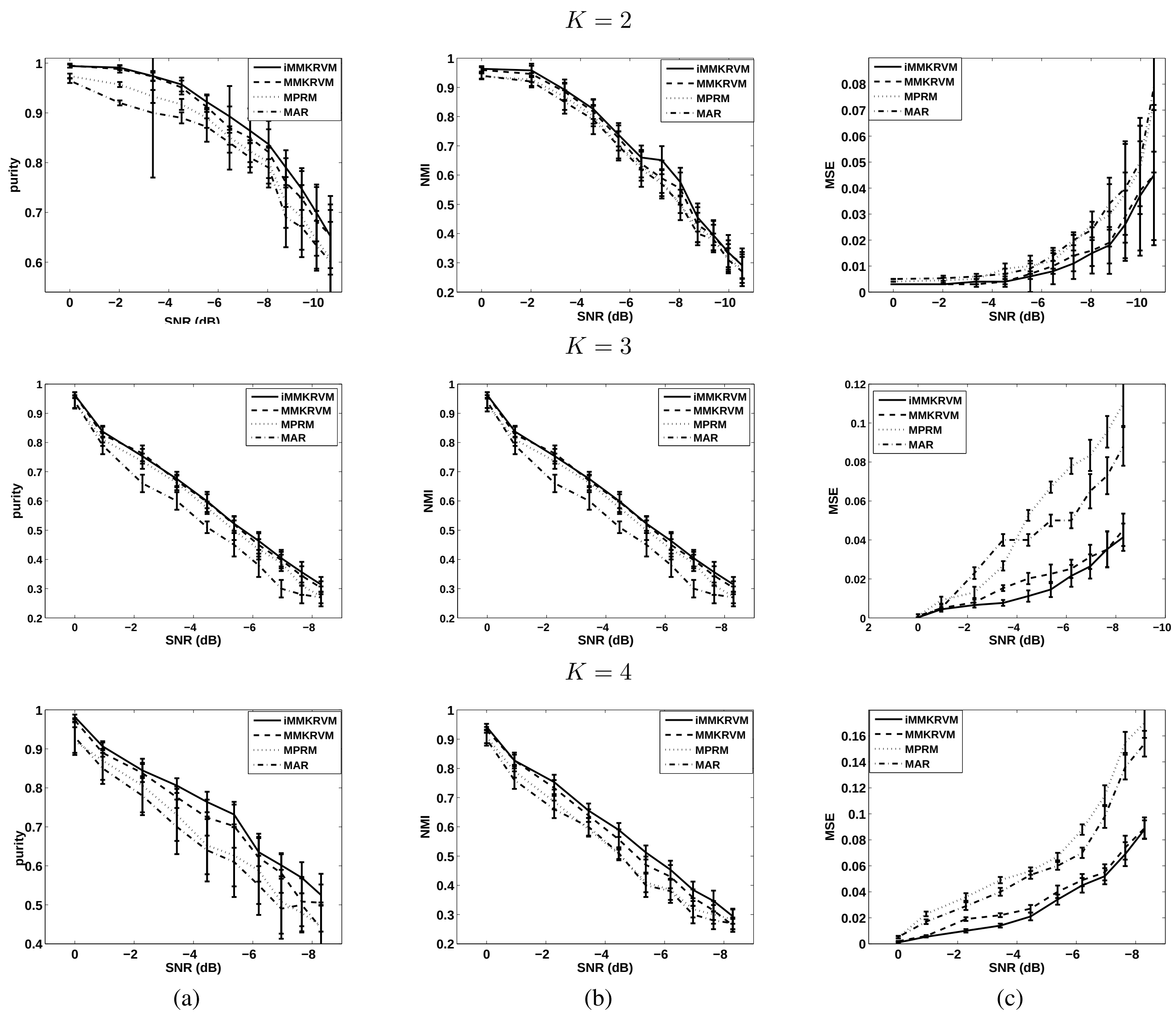


Figure 2. Comparative results for the simulated datasets of Fig. 1. Plots illustrate the three evaluation metrics in terms of various noise levels.

Table II
COMPARATIVE RESULTS OF MIXTURE MODELS FOR THE UCR DATASETS

UCR Dataset	*iMRVM*		*MRVM*		*MPRM*		*MAR*	
	purity	*NMI*	*purity*	*NMI*	*purity*	*NMI*	*purity*	*NMI*
CBF	0.94	0.79	0.85 ± 0.05	0.60 ± 0.09	0.65 ± 0.02	0.38 ± 0.01	0.60 ± 0.03	0.36 ± 0.02
Coffee	0.64	0.06	0.64 ± 0.00	0.06 ± 0.00	0.56 ± 0.00	0.02 ± 0.00	0.57 ± 0.00	0.02 ± 0.00
ECG	0.78	0.35	0.78 ± 0.00	0.35 ± 0.00	0.69 ± 0.00	0.12 ± 0.00	0.72 ± 0.00	0.18 ± 0.00
Face Four	0.69	0.46	0.61 ± 0.03	0.39 ± 0.02	0.40 ± 0.04	0.31 ± 0.02	0.41 ± 0.00	0.29 ± 0.00
Gun Point	0.72	0.16	0.72 ± 0.00	0.16 ± 0.00	0.50 ± 0.00	0.04 ± 0.00	0.55 ± 0.00	0.08 ± 0.00
Synthetic Control	0.76	0.74	0.72 ± 0.02	0.73 ± 0.02	0.73 ± 0.01	0.72 ± 0.01	0.70 ± 0.04	0.69 ± 0.03
Trace	0.75	0.68	0.72 ± 0.02	0.64 ± 0.03	0.53 ± 0.00	0.50 ± 0.00	0.67 ± 0.08	0.61 ± 0.07
Wafer	0.75	0.64	0.75 ± 0.00	0.64 ± 0.00	0.61 ± 0.01	0.00 ± 0.00	0.67 ± 0.04	0.50 ± 0.06

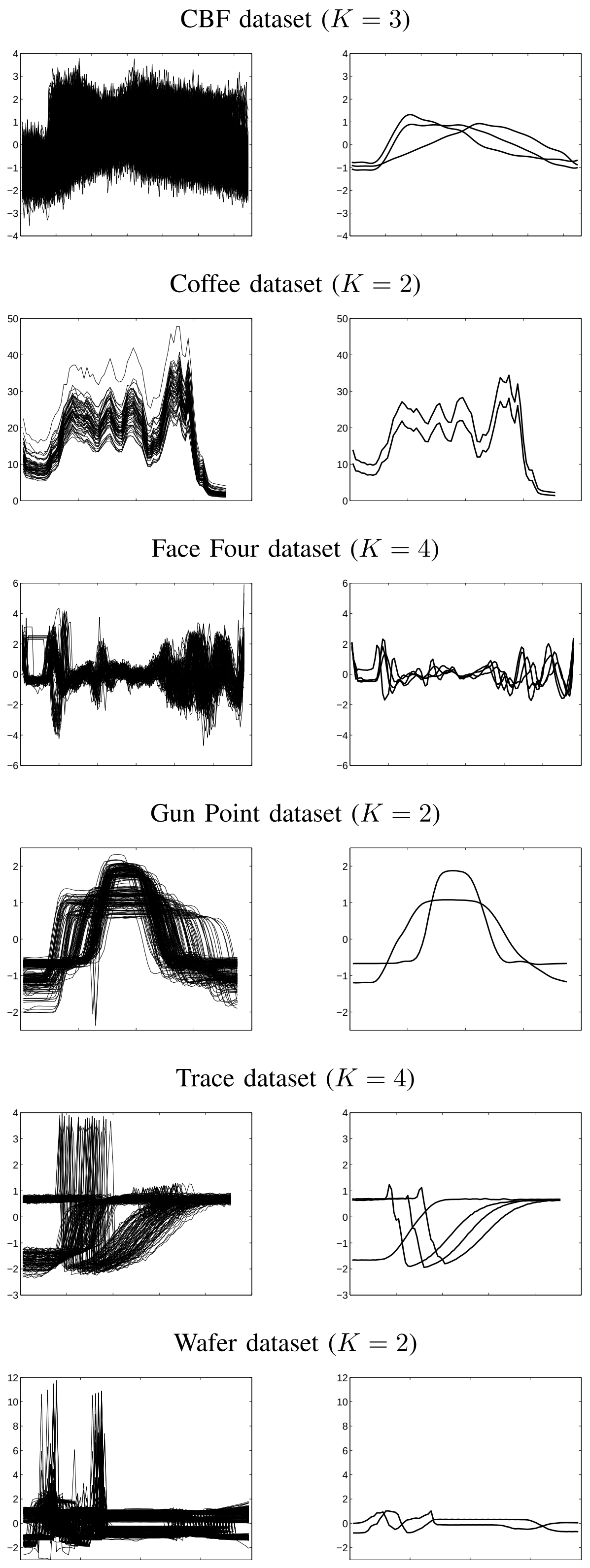


Figure 3. Some examples of the resulting regression function for any component (cluster), as estimated by the proposed method in some UCR datasets.

Gun Point datasets) large kernel width parameters provide a better fit. The proposed method, incorporating the multi-kernel scheme, has the flexibility to automatically adapt to the characteristics of input data samples, thus improving the data fitting capability.

V. Conclusions

In this paper we have proposed a novel regression mixture model, where each mixture component is a multi-kernel RVM regression model. The model is very general and can be used to cluster a set of multidimensional functions, where each function is represented by a set of input-target pairs. The key aspect of the proposed technique lies on the employment of RVMs as components and the exploitation of its superior regression performance to model the data of each latent class. We have also presented a weighted multi-kernel scheme for composing the kernel matrix of each component that offers better fitting capabilities. Learning in the proposed sparse regression mixture model is achieved in terms of a maximum a posteriori (MAP) framework that allows the EM algorithm to be effectively used for estimating the model parameters. This has the advantage of establishing update rules in closed form during the M-step and thus data fitting is computationally efficient. An incremental learning strategy has also been presented that makes the construction of the sparse regression mixture model independent of parameter initialization. Experiments on several datasets for time series clustering demonstrated the ability of the proposed MKRVM mixture to achieve improved clustering performance and robustness compared to other typical regression models.

We are planning to study the performance of the proposed methodology in computer vision applications, such as visual tracking problems and object detection in a video surveillance domain [6], [7], [8]. Another future research direction is to examine the possibility of applying alternative types of sparse priors [17], [18], as well as to make an extensive comparison with other types of mixture models that employ different types of regression components, such as Gaussian Processes. Furthermore, introducing the fully Bayesian framework to our method constitutes an interesting direction for future work, that could also allow for the estimation of the number of clusters.

Acknowledgments

This manuscript is dedicated to the memory of our friend and colleague Professor Nikolaos P. Galatsanos who contributed significantly to the research and preparation of this work. He was a great mentor who continues to provide inspiration to all of his former colleagues

References

[1] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: John Wiley & Sons, Inc., 2000.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] W. DeSarbo and W. Cron, "A maximum likelihood methodology for clusterwise linear regression," *Journal of Classification*, vol. 5, no. 1, pp. 249–282, 1988.

[4] D. Chudova, S. Gaffney, E. Mjolsness, and P. Smyth, "Mixture models for translation-invariant clustering of sets of multi-dimensional curves," in *Proc. of the Ninth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining*, (Washington, DC), pp. 79–88, 2003.

[5] K. Blekas, C. Nikou, N. Galatsanos, and N. V. Tsekos, "A regression mixture model with spatial constraints for clustering spatiotemporal data," *Inter. Journal on Artificial Intelligence Tools*, vol. 17, no. 5, pp. 1023–1041, 2008.

[6] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 375–381, 2003.

[7] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian Learning for Efficient Visual Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1292–1304, 2005.

[8] G. Antonini and J. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp. 1008–1020, 2006.

[9] B. Williams, M. Toussaint, and A. Storkey, "Modelling motion primitives and their timing in biologically executed movements," in *Advances in Neural Information Processing Systems 15*, pp. 1547–1554, 2008.

[10] T. Liao, "Clustering of time series data - a survey," *Pattern Recognition*, vol. 38, pp. 1857–1874, 2005.

[11] P. Smyth, "Clustering sequences with hidden Markov models," in *Advances in Neural Information Processing Systems*, pp. 648–654, 1997.

[12] S. Gaffney and P. Smyth, "Curve clustering with random effects regression mixtures," in *Proc. of the Ninth Intern. Workshop on Artificial Intelligence and Statistics* (C. M. Bishop and B. J. Frey, eds.), 2003.

[13] Y. Xiong and D.-Y. Yeung, "Mixtures of ARMA models for model-based time series clustering," in *IEEE International Conference on Data Mining (ICDM)*, pp. 717–720, 2002.

[14] J. Shi and B. Wang, "Curve prediction and clustering with mixtures of Gaussian process functional regression models," *Statistics and Computing*, vol. 18, pp. 267–283, 2008.

[15] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[16] M. Zhong, "A Variational method for learning Sparse Bayesian Regression," *Neurocomputing*, vol. 69, pp. 2351–2355, 2006.

[17] A. Schmolck and R. Everson, "Smooth Relevance Vector Machine: A smoothness prior extension of the RVM," *Machine Learning*, vol. 68, no. 2, pp. 107–135, 2007.

[18] M. Seeger, "Bayesian Inference and Optimal Design for the Sparse Linear Model," *Journal of Machine Learning Research*, vol. 9, pp. 759–813, 2008.

[19] M. Gonen and E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[20] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B*, vol. 39, pp. 1–38, 1977.

[21] F. Harrell, *Regression Modeling Strategies. With Applications to Linear Models, Logistic Regression and Survival Analysis*. Springer-Verlag New York, Inc., 2001.

[22] S. Gunn and J. Kandola, "Structural modelling with sparse kernels," *Machine Learning*, vol. 48, pp. 137–163, 2002.

[23] M. Girolami and S. Rogers, "Hierarchic bayesian models for kernel learning," in *Intern. Conference on Machine Learning (ICML'05)*, pp. 241–248, 2005.

[24] M. Hu, Y. Chen, and J. Kwok, "Building sparse multiple-kernel svm classifiers," *IEEE Trans. on Neural Networks*, vol. 20, no. 5, pp. 827–839, 2009.

[25] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer-Verlag New York, Inc., 1999.

[26] J. Li and A. Barron, "Mixture density estimation," in *Advances in Neural Information Processing Systems*, vol. 12, pp. 279–285, The MIT Press, 2000.

[27] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton, "SMEM algorithm for mixture models," *Neural Computation*, vol. 12, no. 9, pp. 2109–2128, 2000.

[28] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2001.

[29] E. Keogh, X. Xi, L. Wei, and C. Ratanamahatana, "The ucr time series classification/clustering homepage: www.cs.ucr.edu/∼eamonn/time series data/," 2006.