Knowl Inf Syst (2014) 39:241–264
DOI 10.1007/s10115-013-0704-0

REGULAR PAPER

Sparse regression mixture modeling with the multi-kernel relevance vector machine

Konstantinos Blekas · Aristidis Likas

Received: 1 February 2013 / Revised: 1 August 2013 / Accepted: 15 October 2013 /
Published online: 30 October 2013

Abstract A regression mixture model is proposed where each mixture component is a multi-kernel version of the Relevance Vector Machine (RVM). This mixture model exploits the enhanced modeling capability of RVMs, due to their embedded sparsity enforcing properties. In order to deal with the selection problem of kernel parameters, a weighted multi-kernel scheme is employed, where the weights are estimated during training. The mixture model is trained using the maximum a posteriori approach, where the Expectation Maximization (EM) algorithm is applied offering closed form update equations for the model parameters. Moreover, an incremental learning methodology is also presented that tackles the parameter initialization problem of the EM algorithm along with a BIC-based model selection methodology to estimate the proper number of mixture components. We provide comparative experimental results using various artificial and real benchmark datasets that empirically illustrate the efficiency of the proposed mixture model.

Keywords Relevance vector machines · Mixture models · Sparse prior · Multi-kernel · Incremental EM learning

1 Introduction

Mixture model constitutes a flexible and well-established approach in the case of data sets containing data objects that have been generated from heterogeneous sources [3,21]. 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*.

K. Blekas (✉) · A. Likas
Department of Computer Science and Engineering, University of Ioannina,
P.O. Box 1186, 45110 Ioannina, Greece
e-mail: kblekas@cs.uoi.gr

A. Likas
e-mail: arly@cs.uoi.gr

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 or trucks trajectories [6,5,8,23] and recently to video surveillance problems [1,2,35] and motion recognition [34].

A natural framework for modeling sequential data is through regression mixture models, also known as latent class regression analysis [3,21]. 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 [20]. Among them, Hidden Markov Models [29], polynomial and spline regression models [6,5,11], mixtures of ARMA models [36] and mixtures of Gaussian processes [28] 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 examples [25,27,30,37] 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 use Bayesian inference to obtain sparse models with high generalization by initially employing models with many degrees of freedom on which a heavy tail prior is imposed. 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 [30] assuming a *weighted multikernel* function, ie., each RVM kernel is a weighted combination of basic kernels, and the combination weights are estimated during training [13]. We call this extension a multi-kernel RVM (MKRVM). In the RVM model, the marginal distribution of the observations given the hyperparameters is a Gaussian distribution (see Eq. 10). 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) [7,21] is applied to learn this mixture of multi-kernel RVMs model and fit the input data. This 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 makes the learning process independent of the initialization of model parameters and leads to near-optimal solutions. We also estimate the number of components of the mixture model, and therefore the number of clusters, using the Bayesian information criterion (BIC) [10].

The general learning framework where the proposed regression mixture model can be employed is 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. More specifically, suppose we are given a set of N samples (data objects) $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, where each sample $\boldsymbol{y}_n$ consists of L input-target pairs: $\boldsymbol{y}_n = \{\boldsymbol{x}_n, \boldsymbol{t}_n\} = \{(\boldsymbol{x}_{ni}, t_{ni})\}_{i=1}^{L}$. t_{ni} is the target attribute to be analyzed, while the input $\boldsymbol{x}_{ni}$ can have different formats. An important special case of the above general framework is *time-series clustering*, where the inputs may be either the time instances or d-dimensional vectors containing d past target values, i.e., $\boldsymbol{x}_{ni} = (t_{n,i-d}, t_{n,i-d+1}, \ldots, t_{n,i-1})$. Another special case is *trajectory clustering* where the inputs x_{ni} usually correspond to space coordinates.

We first evaluated the performance of the proposed methodology on the general task of clustering a set of functions using synthetically created data. Then, we tested our method on time-series clustering performance using a variety of artificial 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. 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. Finally, we have experimentally studied the performance of the proposed mixture model on a real problem of clustering trajectories.

As experiments indicate, our method offers both flexibility and robustness and obtains superior modeling solutions. Its modeling power is mainly due to the use of multi-kernel RVM as a mixture component. Multi-kernel RVM is a powerful regressor that exhibits the notable robustness capability of typical RVM that is due to Bayesian regularization. In addition, it effectively tackles the main drawback of typical RVM which is its sensitivity to the choice of the kernel parameters. Although the proposed mixture model is rather complex, the use of Bayesian priors and constraints on the model parameters guide the learning process to avoid overfitting in regression modeling and provide robust regression components. Since the learning process is non-convex and depends on parameter initialization, the proposed incremental training approach also contributes to the quality of the clustering solution. Finally, in the case where we are interested in estimating the true number of clusters, we provide empirical evidence that the simple BIC criterion could provide good estimations in cases where the clusters are well-separated.

In Sect. 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 Sect. 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 Sect. 4 numerical experiments with artificial and real benchmark data sets. Finally, in Sect. 5, we provide conclusions and suggestions for future research.

2 The multi-kernel relevance vector machine

In this section, we present the multi-kernel relevance vector machine that can be applied to model a sample (data object) $\boldsymbol{y}_n$ consisting of L input-target pairs: $\boldsymbol{y}_n = \{\boldsymbol{x}_n, \boldsymbol{t}_n\} = \{(\boldsymbol{x}_{ni}, t_{ni})\}_{i=1}^{L}$. t_{ni} is the target attribute to be analyzed, while the input $\boldsymbol{x}_{ni}$ can have different formats.

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 $\boldsymbol{w} = (w_1, \ldots, w_M)^T$ 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, $\boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \sigma^2 I)$, i.e., errors are not correlated.

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 [15]. Another option is to consider a *kernel* design matrix of size $L \times L$, consisting of L basis functions, $\Phi_n = [\phi(\boldsymbol{x}_{n1}), \ldots, \phi(\boldsymbol{x}_{nL})]$ where $\phi(\boldsymbol{x}_{ni})$ is a vector of L kernel values among $\boldsymbol{x}_{ni}$ and all other inputs, i.e., $\phi(\boldsymbol{x}_{ni}) = (K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{n1}), \ldots, K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{nL}))$. This is achieved by appropriately selecting a kernel function, with the RBF kernel function to be the most commonly used:

$$K(\boldsymbol{x}_{ni}, \boldsymbol{x}_{nk}) = \exp\left(-\frac{\|\boldsymbol{x}_{ni} - \boldsymbol{x}_{ik}\|^2}{2\lambda}\right). \tag{4}$$

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}:

$$\Phi_n = \sum_{s=1}^{S} u_s F_{ns}, \tag{5}$$

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 [12–14,16] 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 Eq. 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.,

$$p(\boldsymbol{t}_n|\boldsymbol{w}, \sigma^2, \boldsymbol{u}) = \mathcal{N}(\boldsymbol{t}_n|\Phi_n \boldsymbol{w}, \sigma^2 I). \tag{6}$$

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 [30]. 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 $\boldsymbol{w} = (w_1, \ldots, w_L)$:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, A^{-1}) = \prod_{i=1}^{L} \mathcal{N}(w_i|0, \alpha_i^{-1}) \tag{7}$$

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 on each hyperparameter α_i:

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{L} \text{Gamma}(\alpha_i|a, b) \propto \prod_{i=1}^{L} \alpha_i^{a-1} e^{-b\alpha_i}. \tag{8}$$

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

$$p(\sigma^{-2}) = \text{Gamma}(\sigma^{-2}|c, d) \propto \sigma^{-2(c-1)} e^{-d\sigma^{-2}}. \tag{9}$$

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* [30], 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 Eq. 6, we can obtain the marginal distribution of target $\boldsymbol{t}_n$ given the model parameters $\theta = \{\boldsymbol{\alpha}, \sigma^2, \boldsymbol{u}\}$, as a zero mean Normal distribution:

$$p(\boldsymbol{t}_n|\theta) = \int p(\boldsymbol{t}_n|\boldsymbol{w}, \sigma^2, \boldsymbol{u}) p(\boldsymbol{w}|\boldsymbol{\alpha}) d\boldsymbol{w} = \mathcal{N}(0, C_n), \tag{10}$$

where the covariance matrix has the form:

$$C_n = \Phi_n A^{-1} \Phi_n^T + \sigma^2 I. \tag{11}$$

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

$$p(\boldsymbol{w}|\boldsymbol{t}_n, \theta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_n, \Sigma_n), \tag{12}$$

with mean and covariance given by

$$\boldsymbol{\mu}_n = \sigma^{-2} \Sigma_n \Phi_n^T \boldsymbol{t}_n, \qquad \Sigma_n = (\sigma^{-2} \Phi_n^T \Phi_n + A)^{-1}. \tag{13}$$

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

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

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

$$\min_{\boldsymbol{u}} \frac{1}{2}\|\boldsymbol{t}_n - \mathcal{F}_n\boldsymbol{u}\|^2 \text{ s.t. } \sum_{s=1}^{S} u_s = 1 \quad \text{and} \quad u_s \geq 0. \tag{14}$$

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

$$\min_{\boldsymbol{u}} \left\{ \frac{1}{2}\boldsymbol{u}^T \mathcal{Z}_n \boldsymbol{u} + \boldsymbol{u}^T \boldsymbol{q}_n \right\}, \quad \text{s.t. } \sum_{s=1}^{S} u_s = 1, u_s \geq 0, \tag{15}$$

where $\mathcal{Z}_n = \mathcal{F}_n^T \mathcal{F}_n$ and $\boldsymbol{q}_n = -\mathcal{F}_n^T \boldsymbol{t}_n$. 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 [22].

3 The mixture of MKRVMs model

Suppose we are given a set of N samples (data objects) $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, 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 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. 10, each component defines a zero mean Normal distribution:

$$p(\boldsymbol{t}_n|\theta_j) = \mathcal{N}(\boldsymbol{t}_n|0, C_{nj}), \tag{16}$$

with

$$C_{nj} = \Phi_{nj} A_j^{-1} \Phi_{nj}^T + \sigma_j^2 I \quad \text{and} \quad \Phi_{nj} = \sum_{s=1}^{S} u_{js} F_{ns}. \tag{17}$$

Moreover, A_j in Eq. 17 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:

$$f(\boldsymbol{t}_n|\Theta) = \sum_{j=1}^{K} \pi_j p(\boldsymbol{t}_n|\theta_j) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj}). \tag{18}$$

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:

$$p(\boldsymbol{\alpha}_j) = \prod_{i=1}^{L} \text{Gamma}(\alpha_{ji}|a_j, b_j) \propto \prod_{i=1}^{L} \alpha_{ji}^{a_j - 1} e^{-b_j \alpha_{ji}}, \tag{19}$$

$$p(\sigma_j^{-2}) = \text{Gamma}(\sigma_j^{-2}|c_j, d_j) \propto \sigma_j^{-2(c_j - 1)} e^{-d_j \sigma_j^{-2}}, \tag{20}$$

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:

$$\begin{aligned} L_{MAP} &= \log p(\boldsymbol{Y}|\Theta) + \log p(\Theta) \\ &= \sum_{n=1}^{N} \log \left\{ \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj}) \right\} + \sum_{j=1}^{K} \left\{ \log p(\boldsymbol{\alpha}_j) + \log p(\sigma_j^{-2}) \right\}, \end{aligned} \tag{21}$$

and use the Expectation–Maximization (EM) algorithm [7] 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:

$$z_{nj} = P(j|\boldsymbol{y}_n, \Theta) = \frac{\pi_j \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj})}{\sum_{j'=1}^{K} \pi_{j'} \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{0}, C_{nj'})}. \tag{22}$$

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:

$$\begin{aligned} Q(\Theta) = &\sum_{n=1}^{N} \sum_{j=1}^{K} z_{nj} \left\{ \log \pi_j - \frac{1}{2} \log |C_{nj}| - \frac{1}{2} \boldsymbol{t}_n^T (C_{nj})^{-1} \boldsymbol{t}_n \right\} \\ &+ \sum_{j=1}^{K} \left\{ \sum_{i=1}^{L} \left\{ a_j \log(\alpha_{ji}) - b_j \alpha_{ji} \right\} + c_j \log(\sigma_j^{-2}) - d_j \sigma_j^{-2} \right\}, \end{aligned} \tag{23}$$

where the quantities z_{nj} have been computed by Eq. 22.

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

$$\hat{\pi}_j = \frac{\sum_{n=1}^{N} z_{nj}}{N}, \tag{24}$$

$$\hat{\alpha}_{ji} = \frac{\sum_{n=1}^{N} z_{nj} + 2a_j}{\sum_{n=1}^{N} z_{nj} \mu_{nji}^2 + \sum_{n=1}^{N} z_{nj} (\Sigma_{nj})_{ii} + 2b_j}, \tag{25}$$

$$\hat{\sigma}_j^2 = \frac{\sum_{n=1}^{N} z_{nj} \|\boldsymbol{t}_n - \Phi_{nj} \boldsymbol{\mu}_{nj}\|^2 + 2d_j}{\sum_{n=1}^{N} z_{nj} (L - \sum_{i=1}^{T} \gamma_{nji}) + 2c_j}. \tag{26}$$

In the above rules we have used the following expressions [30]:

$$\log |\Phi_{nj} A_j^{-1} \Phi_{nj}^T + \sigma_j^2 I| = -\log |\Sigma_{nj}| + \log \sigma_j^2 - \log |A_j|, \tag{27}$$

$$\begin{aligned} \boldsymbol{t}_n^T (\Phi_{nj} A_j^{-1} \Phi_{nj}^T + \sigma_j^2 I)^{-1} \boldsymbol{t}_n &= \frac{1}{\sigma_j^2} \boldsymbol{t}_n^T (\boldsymbol{t}_n - \Phi_{nj} \boldsymbol{\mu}_{nj}) \\ &= \frac{1}{\sigma_j^2} \|\boldsymbol{t}_n - \Phi_{nj} \boldsymbol{\mu}_{nj}\|^2 + \boldsymbol{\mu}_{nj}^T A_j \boldsymbol{\mu}_{nj}, \end{aligned} \tag{28}$$

where

$$\boldsymbol{\mu}_{nj} = \sigma_j^{-2} \Sigma_{nj} \Phi_{nj}^T \boldsymbol{t}_n, \tag{29}$$

$$\Sigma_{nj} = (\sigma_j^{-2} \Phi_{nj}^T \Phi_{nj} + A_j)^{-1}. \tag{30}$$

Note also that in the above equations (Eqs. 25, 26) 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. 14), 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:

$$\min_{\boldsymbol{u}_j} \frac{1}{2} \sum_{n=1}^{N} z_{nj} \|\boldsymbol{t}_n - \Phi_{nj}\boldsymbol{\mu}_{nj}\|^2 = \min_{\boldsymbol{u}_j} \frac{1}{2} \sum_{n=1}^{N} z_{nj} \|\boldsymbol{t}_n - \sum_{s=1}^{S} u_{js} F_{ns}\boldsymbol{\mu}_{nj}\|^2$$
$$= \min_{\boldsymbol{u}_j} \frac{1}{2} \sum_{n=1}^{N} z_{nj} \|\boldsymbol{t}_n - \mathcal{F}_{nj}\boldsymbol{u}_j\|^2 \text{ s.t. } \sum_{s=1}^{S} u_{js} = 1 \text{ and } u_{js} \geq 0, \tag{31}$$

where we have considered only the part of the Q-function (Eq. 23) 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. 15):

$$\min_{\boldsymbol{u}_j} \left\{ \frac{1}{2} \boldsymbol{u}_j^T \mathcal{Z}_j \boldsymbol{u}_j + \boldsymbol{u}_j^T \boldsymbol{q}_j \right\}, \quad \text{s.t. } \sum_{s=1}^{S} u_{js} = 1, u_{js} \geq 0, \tag{32}$$

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 [22].

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. 22). The MKRVM function can be also obtained for each component j as follows:

$$\boldsymbol{w}_j = \left(\sigma_j^{-2} \sum_{n=1}^{N} z_{nj} \Phi_{nj}^T \Phi_{nj} + A_j \right)^{-1} \sigma_j^{-2} \sum_{i=1}^{N} z_{nj} \Phi_{nj}^T \boldsymbol{t}_n. \tag{33}$$

The algorithmic complexity of the proposed methodology depends on the computational cost of the E-step and M-step during the EM learning. The complexity is dominated by the inversion cost $O(NKL^3)$ of the NK covariance matrices C_{nj} (Eq. 17) each of them being an $L \times L$ matrix.

3.1 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. 21). 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 [19,31,32]. 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 $z_{nj} = 1$.

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

$$f_k(\boldsymbol{t}_n|\Theta_k) = \sum_{j=1}^{k} \pi_j \, p(\boldsymbol{t}_n|\theta_j). \tag{34}$$

Also, we denote as:

$$f_k^{-j}(\boldsymbol{t}_n|\Theta_k^{-j}) = f_k(\boldsymbol{t}_n|\Theta_k) - \pi_j \, p(\boldsymbol{t}_n|\theta_j), \tag{35}$$

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:

$$f_{k+1}(\boldsymbol{t}_n|\Theta_{k+1}) = f_k^{-j^*}(\boldsymbol{t}_n|\Theta_k^{-j^*}) + \pi_{j^*}{}^{new} \, p(\boldsymbol{t}_n|\theta_{j^*}) + \pi_{k+1} \, p(\boldsymbol{t}_n|\theta_{k+1}). \tag{36}$$

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 (e.g., 20 % in our experiments) 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 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. This can be seen very convenient for introducing model order selection criteria and terminating the evolution of learning: stop training when the insertion of a new component does not offer any significant improvement of the (penalized) likelihood function.

3.2 Model order selection

The problem of selecting a statistical model of correct order is fundamental in statistical learning. In mixture models, this corresponds to the problem of choosing the proper number

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 Θ_{k+1}.
8: $k = k + 1$.
9: **end while**

of mixture components K. Among the various methods for model order selection, in this study we have used the Bayesian information criterion (BIC) [26] which provides an approximation of Bayes factors and has been successfully applied in a number of applications, see for example [10,33]. If we recall that $L(.)$ is the log-likelihood function as defined in Eq. 21 and let Θ_k be the maximum likelihood estimate of the k order model, then the BIC value is given by:

$$\text{BIC} = -2L(\Theta_k) + G\log(N). \tag{37}$$

The quantity G is the total number of model parameters, where in our case is $G = (k-1) + kT + k + k(S-1) = k(T+S+1) - 1$, since there are four kinds of parameters $\{\boldsymbol{\pi}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{u}\}$ and the constraints $\sum_j \pi_j = 1$, $\sum_s u_{js} = 1\ \forall j = 1, \ldots, K$. In this way, BIC includes a penalty term that depends on the number of model parameters. This term penalizes complex models with many parameters and thus counterbalances the negative log-likelihood term which decreases monotonically with the number of parameters.

Finally, it must be noted that, as mentioned before, the proposed incremental scheme for building the mixture model of the MKRVM components is very convenient, since it provides the parameters Θ_k and the log-likelihood $L(\Theta_k)$ of all successive models $[\Theta_1, \ldots, \Theta_K]$ simultaneously. Thus, we execute the learning procedure only once by setting a large value to the number of components K and then we select the mixture with minimum BIC score.

4 Experimental results

The proposed mixture model has been evaluated using a variety of artificial datasets and real benchmarks. We have considered both the general case where the samples $\boldsymbol{y}_n$ consist of input–output pairs $(\boldsymbol{x}_n, \boldsymbol{t}_n)$ (i.e., inputs are given), as well as the tasks of time-series and trajectory clustering. In all experiments 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 $k_s = [0.1, 0.2, \ldots, 1.0]$ (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. In the case

of time-series clustering we compared our method with two common regression mixture models (see Sect. 4.2): the polynomial regression mixture model (*MPRM*) and the mixture of autoregressive (*MAR*) model.

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

$$\text{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}, \tag{38}$$

where

$$I(\Omega, \mathcal{C}) = \sum_k \sum_j P(\omega_k, c_j) \log \frac{P(\omega_k, c_j)}{P(\omega_k) P(c_j)} \tag{39}$$

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) \tag{40}$$

$$H(\mathcal{C}) = -\sum_k P(c_k) \log P(c_k). \tag{41}$$

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).

- *mean square error* (MSE) between the original series $\{\boldsymbol{r}_j,\ j = 1, \ldots, K\}$ and the estimated mean functional curves $\bar{\boldsymbol{t}}_j$ after convergence calculated as:

$$\text{MSE} = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{L} \|\boldsymbol{r}_j - \bar{\boldsymbol{t}}_j\|^2, \tag{42}$$

where

$$\bar{\boldsymbol{t}}_j = \frac{\sum_{n=1}^{N} z_{nj} \Phi_{nj} \boldsymbol{\mu}_{nj}}{\sum_{n=1}^{N} z_{nj}}. \tag{43}$$

This evaluation criterion was used in the case of artificial datasets, since we are aware of the generative series of each cluster.

4.1 Clustering artificial functional data

At first, we have evaluated our method to a synthetic dataset created by a set of functional data sources. In particular, we have used a pool of K functional parametric forms $f_j(\boldsymbol{x}; \vartheta_j)$ and a grid of L input data points $\{\hat{\boldsymbol{x}}_i, i = 1, \ldots, L\}$. In order to generate the n-th sample (data object), at first a functional form f_j was selected and its parameters ϑ were specified by adding noise ϵ to the parameter vector ϑ_j. In this way a function $g_n(\boldsymbol{x}; \vartheta) = f_j(\boldsymbol{x}; \vartheta_j + \epsilon)$ is produced. Next, we add noise to the points $\{\hat{\boldsymbol{x}}_i\}$ to generate the inputs $\{\boldsymbol{x}_{ni} = \hat{\boldsymbol{x}}_i + \epsilon\}$. Then, the targets t_{ni} are computed as $t_{ni} = g_n(\boldsymbol{x}_{ni})$.

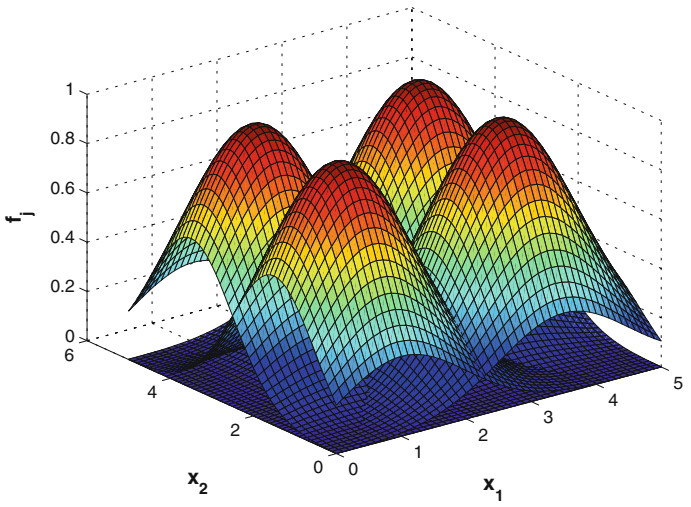

Fig. 1 The four (4) two-dimensional RBF functions (with different centers) used for constructing synthetic datasets with $K = \{2, 3, \text{or } 4\}$ clusters

In our study, we have used K RBF functions with different centers $\boldsymbol{m}_j$, $(j = 1, \ldots, K)$ and constant radius (equal to 1):

$$f_j(\boldsymbol{x}; \boldsymbol{m}_j) = \exp(-0.5|\boldsymbol{x} - \boldsymbol{m}_j|^2). \tag{44}$$

defined on a 2-dimensional (15×15) grid in the domain $[0, 5] \times [0, 5]$. Figure 1 shows the four RBF functions f_j, evaluated on the corresponding $L = 225$ grid input points.

Three different data sets have been studied using $K = \{2, 3, 4\}$ RBF functions, respectively. A number of 100 noisy copies per functional class were generated (as previously described) assuming a specific noise level (variance), thus creating a dataset with $N = 100 \times K$ samples (data objects). For every noise level, we generated 30 different datasets and we calculated the mean value and the standard deviation of three performance criteria: purity, NMI and MSE. The obtained results are shown in Fig. 2 for the various level of noise. As it is obvious, the proposed method manages to distinguish well among K functions, especially in cases with low level of noise. When the noise grows the results becomes lower due to the significant overlapping among the K RBF functions.

Additional experiments have been performed in order to evaluate the BIC-based model selection methodology. In particular, for each set (with $K = 2$, 3 and 4 clusters, respectively) and noise level (small, medium, large), a group of 50 datasets were stochastically generated. The BIC-based methodology was executed and we measured the relative frequency of the estimated number of mixture components $\hat{K}$. Note that for each dataset the proposed incremental learning scheme of the MMKRVM was executed only once until a model with $K_{max} = 10$ components had been constructed. Figure 3 presents the histogram results, i.e., plots of the relative frequencies of the obtained estimates $\hat{K}$ found. It is apparent that the true number of clusters K can be accurately deduced most of the times in the low noise experiments. As noise levels increase, estimation performance deteriorates as K increases.

4.2 Time-series clustering

In the task of time-series clustering, we consider each sample as a sequence of real observations measured at L successive time instances that correspond to the target values t_{ni}. At each

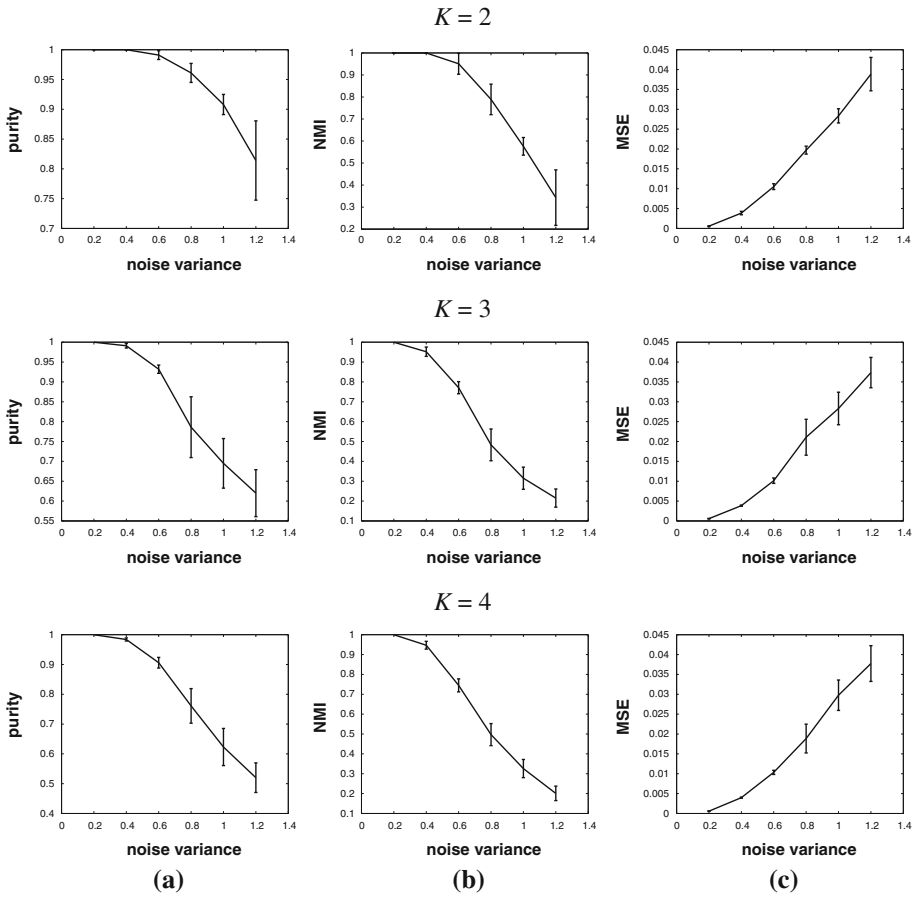

Fig. 2 Performance results of the proposed method on the synthetic functional datasets. Plots illustrate the three evaluation metrics in terms of various noise levels

time instance, the input $\boldsymbol{x}_{ni}$ is a d-dimensional vector that describes the d previous target values, i.e., $\boldsymbol{x}_{ni} = (t_{n,i-d}, t_{n,i-d+1}, \ldots, t_{n,i-1})$. During all experiments, we have considered inputs of length $d = 10$. It must be noted that the objective of our experiments is to evaluate the clustering ability of the proposed method. In the experiments with time series, we consider each time series as an object to be clustered, and we are not interested in identifying possible subsequences in every time series [17,24].

In this case of time-series clustering, 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.,

$$t_{ni} = \sum_{l=0}^{p} w_{jl} x_{ni}^{l}, \tag{45}$$

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

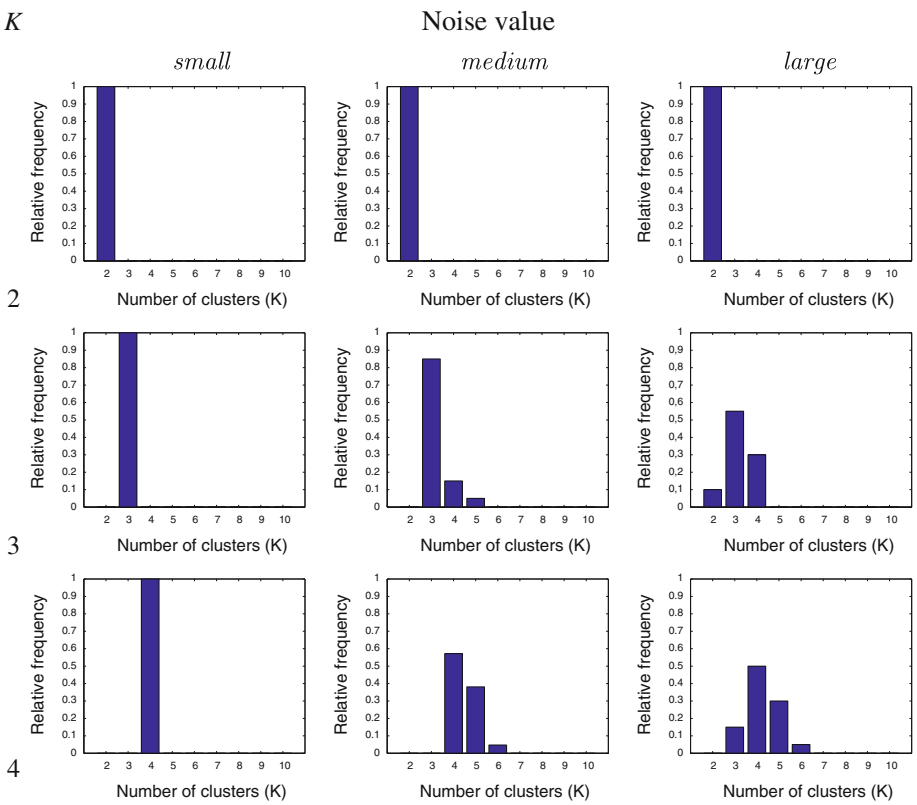

Fig. 3 Results from the use of the BIC criterion on the synthetic functional dataset as histograms of the estimated number of clusters for three noise levels

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.,

$$t_{ni} = w_{j0} + \sum_{l=1}^{p} w_{jl} t_{n,i-l}. \tag{46}$$

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 [5,11,36], where we follow the typical sample-based initialization strategy described in Sect. 3.1.

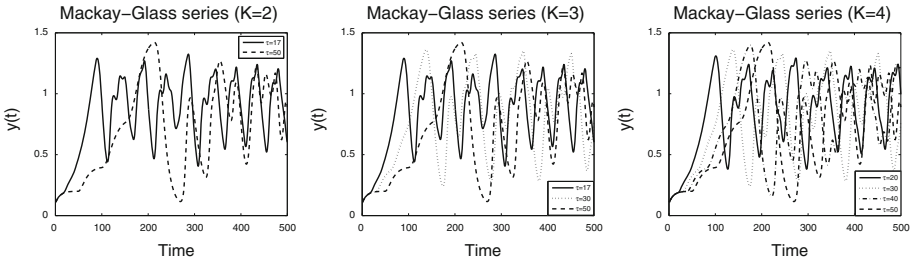

Fig. 4 Three sets with **a** $K = 2$, **b** $K = 3$ and **c** $K = 4$ Mackey–Glass series used for generating artificial datasets

4.2.1 Experiments with artificial data

At first, we have made a series of experiments with artificial datasets for evaluating the performance of our method on time-series clustering. 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:

$$r(t+1) = r(t) + \delta\left(0.2\frac{r(t-\tau/\delta)}{1+r(t-\tau/\delta)^{10}} - 0.1r(t)\right), \tag{47}$$

where the step size 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. 4. A number of 100 noisy copies of the original curve per class were generated using various levels of noise. Similar to the previous case study, 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 5 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, 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.

Finally, we have made additional experiments in order to evaluate the BIC-based model selection methodology following the same strategy as described previously. In particular, for each of the three Mackey–Glass sets (Fig. 1), we created 50 different datasets by adding noise to the function values (three levels of noise: small, medium and large), where we measured the relative frequency of the estimated number of components. Figure 6 presents the obtained histogram results, i.e., plots of the relative frequencies of the number of clusters found for the three Mackey–Glass sets of Fig. 4 and three noise levels. It is clear that the true number of clusters K can be accurately deduced most of the times in the low noise experiments. As noise levels increase, estimation performance deteriorates as K increases. Nevertheless, it

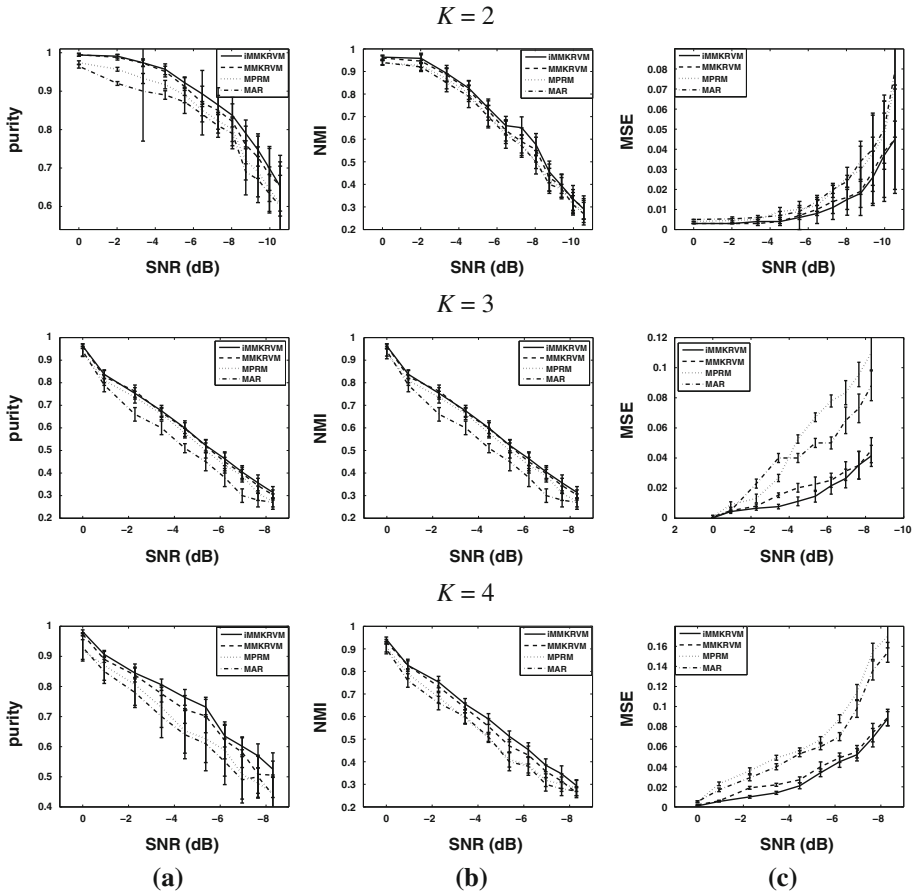

Fig. 5 Comparative results for the simulated datasets of Fig. 4. Plots illustrate the three evaluation metrics in terms of various noise levels

must be underlined that in the case of $K = 3$ and 4, the time-series exhibit very high overlap (see Fig. 4b, c); thus, it is difficult to identify separated clusters.

4.2.2 Experiments with real benchmarks

Further experiments have been conducted using various real datasets, obtained from the UCR time-series data collection [9,18], where the ground truth is known. In Table 1, we present a summary of thirteen (13) UCR datasets we have used in our study. The results using two evaluation metrics, purity and NMI, are shown in Tables 2 and 3, respectively, for the two versions of the proposed mixture of MKRVMs and the other two regression mixture models. Since the proposed incremental learning approach (iMMKRVM) does not depend on the initialization, we show only the result of a single run. For the rest three methods (MMKRVM, MPRM, MAR), we provide the mean value and the standard deviation of each measure (for 30 trials). As can be observed, the performance of the proposed mixture of MKRVMs is obviously superior and in many cases the difference is quite noticeable.

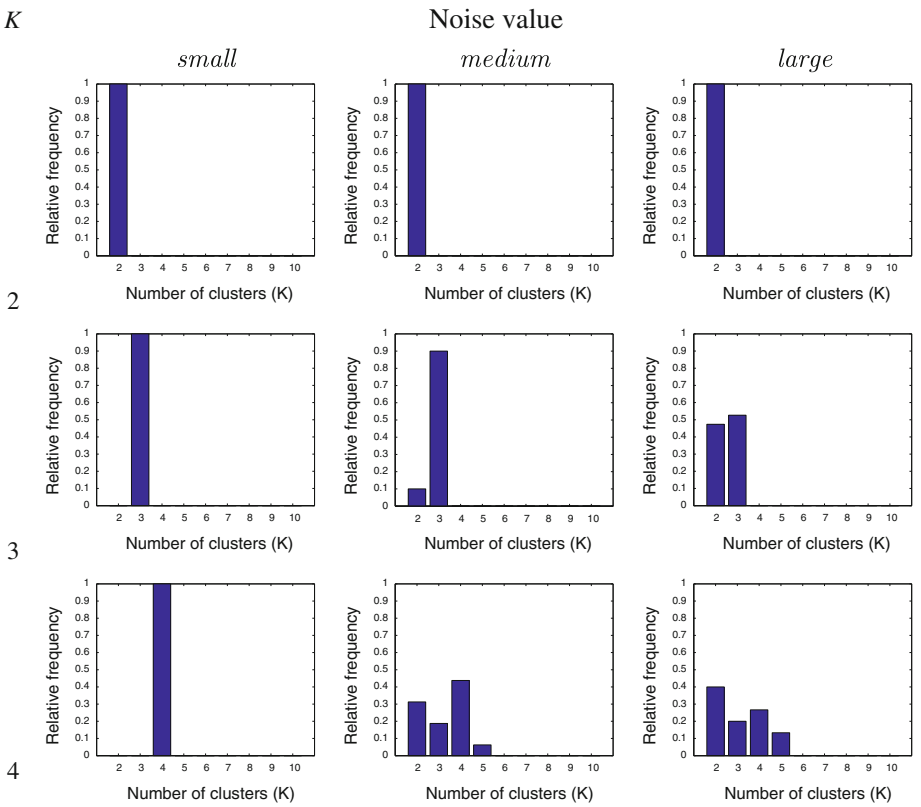

Fig. 6 Histograms of the estimated number of clusters by applying the BIC model selection method to the mixture of MKRVMs for three noise levels added to the three sets of Mackey Glass series (Fig. 4) with $K = 2, 3$ and 4 clusters, respectively

Table 1 Description of the 13 UCR datasets used in our experimental study

Dataset	# classes (K)	Size (N)	Dimension (T)
CBF	3	930	128
Coffee	2	56	286
Diatom size reduction	3	467	166
ECG	2	200	96
Face four	4	112	350
Gun point	2	200	150
Sony AIBO robot I	2	621	70
Sony AIBO robot II	2	1,018	65
Star light curves	3	1,000	1,024
Symbols	6	1,020	398
Synthetic control	6	600	60
Trace	4	200	275
Wafer	2	1,000	152

Table 2 Comparative results (purity metric) of regression mixture models for the UCR datasets

UCR dataset	iMMKRVM	MMKRVM	MPRM	MAR
CBF	0.94	0.85 (0.05)	0.65 (0.02)	0.60 (0.03)
Coffee	0.64	0.64 (0.00)	0.56 (0.00)	0.57 (0.00)
Diatom size reduction	0.95	0.89 (0.02)	0.78 (0.00)	0.59 (0.03)
ECG	0.78	0.78 (0.00)	0.69 (0.00)	0.72 (0.00)
Face four	0.69	0.61 (0.03)	0.40 (0.04)	0.41 (0.00)
Gun point	0.72	0.72 (0.00)	0.50 (0.00)	0.55 (0.00)
Sony AIBO Robot I	0.93	0.92 (0.01)	0.92 (0.01)	0.91 (0.00)
Sony AIBO Robot II	0.81	0.81 (0.00)	0.73 (0.00)	0.92 (0.01)
Star light curves	0.74	0.74 (0.00)	0.74 (0.00)	0.57 (0.00)
Symbols	0.81	0.75 (0.03)	0.70 (0.06)	0.60 (0.10)
Synthetic control	0.76	0.72 (0.02)	0.73 (0.01)	0.70 (0.04)
Trace	0.75	0.72 (0.02)	0.53 (0.00)	0.67 (0.08)
Wafer	0.75	0.75 (0.00)	0.61 (0.01)	0.67 (0.04)

Table 3 Comparative results (NMI metric) of regression mixture models for the UCR datasets

UCR dataset	iMMKRVM	MMKRVM	MPRM	MAR
CBF	0.79	0.60 (0.09)	0.38 (0.01)	0.36 (0.02)
Coffee	0.06	0.06 (0.00)	0.02 (0.00)	0.02 (0.00)
Diatom size reduction	0.87	0.76 (0.04)	0.83 (0.01)	0.55 (0.02)
ECG	0.35	0.35 (0.00)	0.12 (0.00)	0.18 (0.00)
Face four	0.46	0.39 (0.02)	0.31 (0.02)	0.29 (0.00)
Gun point	0.16	0.16 (0.00)	0.04 (0.00)	0.08 (0.00)
Sony AIBO Robot I	0.65	0.61 (0.02)	0.63 (0.01)	0.59 (0.00)
Sony AIBO Robot II	0.35	0.35 (0.00)	0.16 (0.00)	0.62 (0.02)
Star light curves	0.62	0.62 (0.00)	0.58 (0.00)	0.23 (0.00)
Symbols	0.74	0.74 (0.02)	0.75 (0.05)	0.58 (0.07)
Synthetic control	0.74	0.73 (0.02)	0.72 (0.01)	0.69 (0.03)
Trace	0.68	0.64 (0.03)	0.50 (0.00)	0.61 (0.07)
Wafer	0.64	0.64 (0.00)	0.00 (0.00)	0.50 (0.06)

Figure 7 illustrates the mean mixture regression functions as estimated by the proposed iMMKRVM model (according to Eq. 43) 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, by the choice of the 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 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.

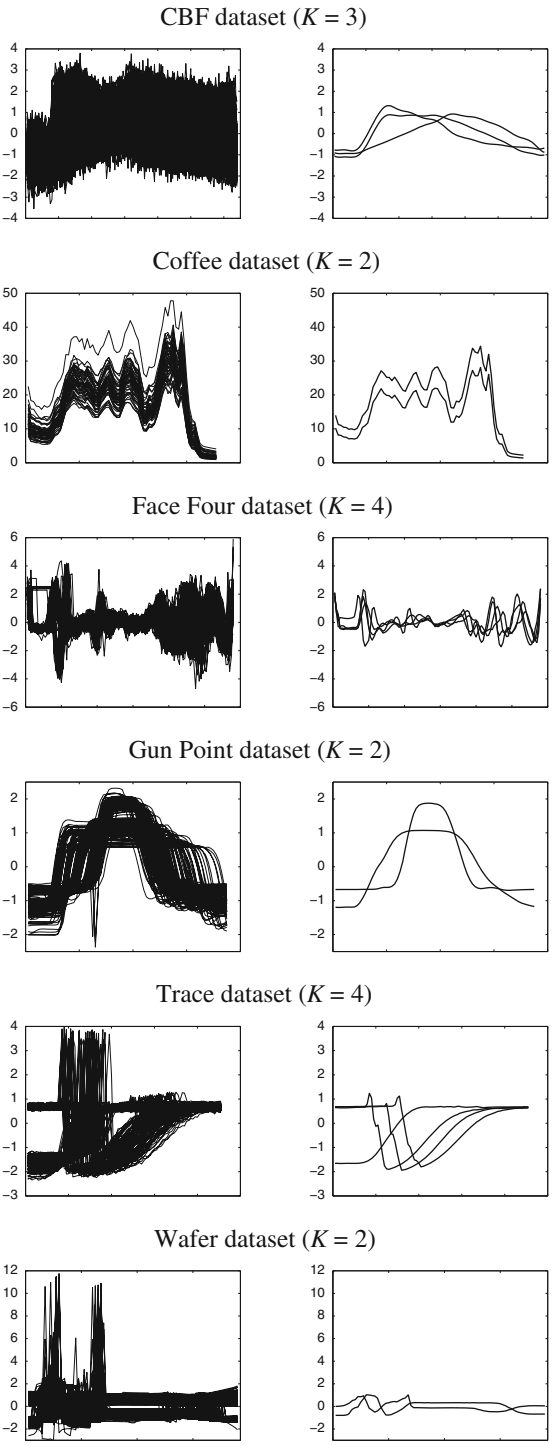

Fig. 7 Some examples of the resulting regression function for any component (cluster) as estimated by the proposed method in some UCR datasets

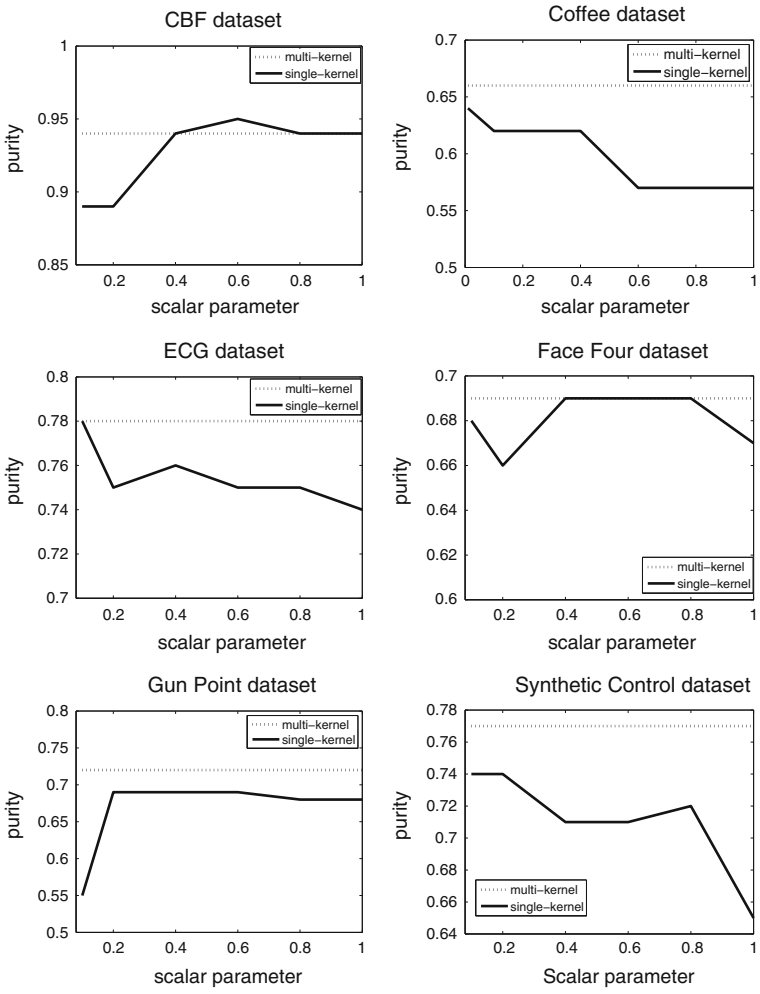

Fig. 8 Impact of the kernel scaling parameter λ on the clustering performance. These are plots of the purity evaluation metric in terms of value of λ as a percentage of the total samples variance of six different UCR datasets

In this direction, we have made additional experiments to study the influence of the multi-kernel scheme on the performance of the clustering process. In particular, we have considered a mixture model with *single kernel* RVMs where we have assumed a kernel design matrix with constant value for the scaling parameter λ (Eq. 4). The results are shown in Fig. 8 that plots the performance of the mixture model of single-kernel RVMs in terms of λ, while the results for the proposed mixture model of MKRVMs are shown with dotted lines.

The results indicate that the selection of a proper value of λ is an important issue for the performance of the clustering procedure. Moreover, there are some cases where, using the single-kernel scheme, we are not able to determine a value for λ providing better performance compared with the multi-kernel case. It must be noted here that we have also studied other kernel design matrices, such as wavelet-based kernel, without obtaining better results. Our empirical results indicate that the employment of the Gaussian multi-kernel framework is a good choice for regression mixture modeling.

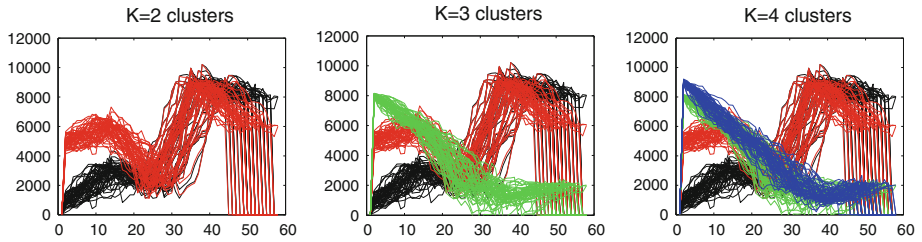

Fig. 9 The truck trajectories shown as raw data used in our experiments. Three datasets are shown containing trajectories of $K = 2$, $K = 3$ and $K = 4$ clusters, respectively

Table 4 Clustering performance of our method on the trucks trajectory dataset under different number of clusters

Problem: number of clusters					
$K = 2$		$K = 3$		$K = 4$	
Purity	NMI	Purity	NMI	Purity	NMI
1.0	1.0	1.0	1.0	0.87	0.76

4.3 Clustering real trajectories

We have also studied our method with a real trajectory dataset where the ground truth is known. It consists of GPS-tracked positions of 50 trucks transporting concrete in the area of Athens between August and September 2002 [23], and the goal is to discover complex mobility patterns. From the original raw data, smaller trajectories were created by splitting the recordings of a truck in subsets if there was a temporal gap between two consecutive recordings larger than 15 min. Each trajectory y_n shows a round trip performed by a truck consisting of the target values t_n and the geographical coordinates x_n. A more detailed description about this dataset can be found in [23].

There are four kinds of possible directions of trips performed by tracks that were manually discovered. In our study we have used a subset of this dataset consisting of 50 trajectories per each cluster of length $L = 58$. Following the experimental methodology on this dataset of the original work described in [23], we conducted a series of experiments using different portions of the trucks trajectories containing two ($K = 2$), three ($K = 3$) or four ($K = 4$) clusters. Figure 9 shows these three (overlap) sets of trucks trajectories used in our study. The obtained results (purity and NMI evaluation metrics) are shown in Table 4. The proposed incrementally constructed mixture of MKRVMs showed excellent behavior in the case of $K = 2$ and 3 clusters, while the performance somehow deteriorates for $K = 4$ clusters, since as shown in Fig. 9, there is a significant overlapping between trajectories of the $3rd$ and the 4th cluster (green and blue colored trajectories). In our experiments with this dataset, we have also studied the BIC model selection criterion. According to the results, the estimated number of components was much higher than the real value of the clusters K. There are two main reason for this behavior: first, due to the structure of data and the existence of subclusters and secondly because of the relevant small number of data per cluster (note that BIC is an accurate measure only in the limit; thus, it generally requires a lot of data for making accurate decisions). As a result, the BIC criterion fails leading to an over-estimation of the true number of clusters.

5 Conclusions

In this work, we presented a powerful 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. Finally, we have considered the BIC criterion for choosing the number of components in the regression mixture model, and thus estimating the structure of the model. Clustering experiments on several datasets using simulated functional data, time-series and trajectories, demonstrated the ability of the proposed MKRVM mixture to achieve improved clustering performance and robustness compared with 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 [1,2,35]. Another future research direction is to examine the possibility of applying alternative types of sparse priors [25,27]. Furthermore, instead of using BIC for model selection, the fully Bayesian mixture of multi-kernel RVMs could be defined and trained providing an alternative methodology for the estimation of the number of clusters.

Acknowledgments This paper substantially improves and extends our previous work presented in [4]. 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.

References

1. Alon J, Sclaroff S, Kollios G, Pavlovic V (2003) Discovering clusters in motion time-series data. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 375–381
2. Antonini G, Thiran J (2006) Counting pedestrians in video sequences using trajectory clustering. IEEE Trans Circuits Syst Video Technol 16(8):1008–1020
3. Bishop C (2006) Pattern recognition and machine learning. Springer, Berlin
4. Blekas K, Likas A (2012) The mixture of multi-kernel relevance vector machines model. In: International Conference on Data Mining (ICDM), pp 111–120
5. Blekas K, Nikou C, Galatsanos N, Tsekos NV (2008) A regression mixture model with spatial constraints for clustering spatiotemporal data. Int J Artif Intell Tools 17(5):1023–1041
6. Chudova D, Gaffney S, Mjolsness E, Smyth P (2003) Mixture models for translation-invariant clustering of sets of multi-dimensional curves. In: Proceedings of the Ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Washington, pp 79–88
7. Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. J Royal Stat Soc B 39:1–38
8. DeSarbo W, Cron W (1988) A maximum likelihood methodology for clusterwise linear regression. J Classif 5(1):249–282
9. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. In PVLDB, pp 1542–1552

10. Fraley C, Raftery AE (1998) Bayesian regularization for normal mixture estimation and model-based clustering. Comput J 41:578–588
11. Gaffney S, Smyth P (2003) Curve clustering with random effects regression mixtures. In: Bishop CM, Frey BJ (eds) Proceedings of the ninth international workshop on artificial intelligence and statistics
12. Girolami M, Rogers S (2005) Hierarchic bayesian models for kernel learning. In: International conference on machine learning (ICML'05), pp 241–248
13. Gonen M, Alpaydin E (2011) Multiple kernel learning algorithms. J Mach Learn Res 12:2211–2268
14. Gunn S, Kandola J (2002) Structural modelling with sparse kernels. Mach Learn 48:137–163
15. Harrell F (2001) Regression modeling strategies. With applications to linear models, logistic regression and survival analysis. Springer, New York
16. Hu M, Chen Y, Kwok J (2009) Building sparse multiple-kernel SVM classifiers. IEEE Trans. Neural Netw 20(5):827–839
17. Keogh E, Lin J, Truppel W (2005) Clustering of time series subsequences is meaningless: implications for past and future research. Knowl Inf Syst KAIS 2:154–177
18. Keogh E, Xi X, Wei L, Ratanamahatana C (2006) The ucr time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/timeseriesdata/
19. Li J, Barron A (2000) Mixture density estimation. In: Advances in neural information processing systems, Vol 12. The MIT Press, Cambridge, pp 279–285
20. Liao T (2005) Clustering of time series data: a survey. Patt Recognit 38:1857–1874
21. McLachlan G, Peel D (2000) Finite mixture models. Wiley, New York
22. Nocedal J, Wright SJ (1999) Numerical optimization. Springer, New York
23. Pelekis N, Kopanakis I, Kotsifakos E, Frentzos E, Theodoridis Y (2011) Clustering uncertain trajectories. Knowl Inf Syst KAIS 28:117–147
24. Rakthanmanon T, Campana B et al (2013) Addressing big data time series: mining trillions of time series subsequences under dynamic time warping. ACM Trans Knowl Discov Data 7(3):1–31
25. Schmolck A, Everson R (2007) Smooth relevance vector machine: a smoothness prior extension of the RVM. Mach Learn 68(2):107–135
26. Schwarz C (1978) Estimating the dimension of a model. Ann Stat 6:461–464
27. Seeger M (2008) Bayesian inference and optimal design for the sparse linear model. J Mach Learn Res 9:759–813
28. Shi J, Wang B (2008) Curve prediction and clustering with mixtures of Gaussian process functional regression models. Stat Comput 18:267–283
29. Smyth P (1997) Clustering sequences with hidden Markov models. In: Advances in neural information processing systems, pp 648–654
30. Tipping M (2001) Sparse Bayesian learning and the relevance vector machine. J Mach Learn Res 1:211–244
31. Ueda N, Nakano R, Ghahramani Z, Hinton G (2000) SMEM algorithm for mixture models. Neural Comput 12(9):2109–2128
32. Vlassis N, Likas A (2001) A greedy EM algorithm for Gaussian mixture learning. Neural Process Lett 15:77–87
33. Wasserman L (2000) Bayesian model selection and model averaging. J Math Psychol 44(1):92–107
34. Williams B, Toussaint M, Storkey A (2008) Modelling motion primitives and their timing in biologically executed movements. In: Advances in neural information processing systems, vol 15, pp 1547–1554
35. Williams O, Blake A, Cipolla R (2005) Sparse Bayesian learning for efficient visual tracking. IEEE Trans. Pattern Anal Mach Intell 27(8):1292–1304
36. Xiong Y, Yeung D-Y (2002) Mixtures of ARMA models for model-based time series clustering. In: IEEE international conference on data mining (ICDM), pp 717–720
37. Zhong M (2006) A variational method for learning sparse Bayesian regression. Neurocomputing 69:2351–2355

Author Biographies

Konstantinos Blekas received the Diploma degree in Electrical Engineering in 1993 and the Ph.D. degree in Electrical and Computer Engineering in 1997, both from the National Technical University of Athens. He has been on the Faculty of the Department of Computer Science and Engineering, University of Ioannina, Greece since 2002, where he is currently an Assistant Professor. His research interests include Artificial Intelligence, Machine Learning and Pattern Recognition with applications to Computer Vision, Robotics, Bioinformatics and Medical data analysis.

Aristidis Likas received the Diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering both from the National Technical University of Athens, Greece. Since 1996, he has been with the Department of Computer Science, University of Ioannina, Greece, where he is currently a Professor. His research interests include machine learning, data mining, neural networks, multimedia data analysis and bioinformatics.