Sparse Multivariate Gaussian Mixture Regression

Luis Weruaga, *Senior Member, IEEE*, and Javier Vía, *Senior Member, IEEE*

***Abstract*—Fitting a multivariate Gaussian mixture to data represents an attractive, as well as challenging problem, in especial when sparsity in the solution is demanded. Achieving this objective requires the concurrent update of all parameters (weight, centers, and precisions) of all multivariate Gaussian functions during the learning process. Such is the focus of this paper, which presents a novel method founded on the minimization of the error of the generalized logarithmic utility function (GLUF). This choice, which allows us to move smoothly from the mean square error (MSE) criterion to the one based on the logarithmic error, yields an optimization scenario that resembles a "locally" convex problem and can be solved with a quasi-Newton method. The GLUF framework also facilitates the comparative study between both extremes, concluding that the classical MSE optimization is not adequate for the task. The performance of the proposed novel technique is demonstrated on simulated as well as realistic scenarios.**

***Index Terms*—Gaussian function mixture, function approximation, regression, logarithmic utility function, sparsity.**

I. Introduction

The Gaussian function has many appealing properties, such as in universal function approximation [1], minimum time–frequency dispersion [2], and outcome of the *central limit theorem*. Fitting a Gaussian function mixture (GFM) on data has proven very useful in selected problems of signal and information processing [3]–[10]. Despite function approximation and data regression with GFM counts with solid foundations [11]–[14], several questions still remain open, such as the update of the Gaussian variance, or the development of mechanisms that promote sparsity in the mixture.

The d-variate Gaussian function is defined by

$$\varphi(\mathbf{x}) = w \exp\left(-(\mathbf{x}-\mathbf{c})^T \mathbf{P}(\mathbf{x}-\mathbf{c})\right) \tag{1}$$

where ^{T} denotes transpose, $\mathbf{P}$ is the $d \times d$ symmetric precision matrix, which is positive definite $\mathbf{P} \succ 0$, $\mathbf{c}$ the d-dimensional center, and w is the amplitude of the Gaussian. Let $\mathbf{x}$ represent a d-dimensional point and $y(\mathbf{x})$ its reference value, the problem we study in this paper is namely to approximate or to represent the function of reference $y(\mathbf{x})$ at given points $\mathbf{x}$ as a mixture of K Gaussian functions

$$f(\mathbf{x}) = \sum_{k=1}^{K} \varphi_k(\mathbf{x}) \tag{2}$$

where $\varphi_k(\mathbf{x})$ is the kth Gaussian function in the mixture. Fig. 1 illustrates this problem with a two-dimensional toy scenario.

Because the precision is defined as a matrix, the multivariate Gaussian (1) possesses large approximation capabilities. In turn, the abundance of degrees of freedom, especially in large dimensional input spaces, increases the risk of overfitting. Therefore, in order to limit the capacity of the multivariate Gaussian kernel, a single-value precision is often the standard choice [11], [13], [21]. Despite this simplification, the estimation of all Gaussian kernel parameters, including the precision, is a challenging problem that remains largely unsolved. For instance, in the so-called radial basis functions (RBF) [11], [12], [14], the precision of the Gaussian functions cannot be easily updated with gradient descent [14], [15], as the optimization problem is non-convex. In support vector regression (SVR) and other sparse kernel modelling methods [13], the kernel centers are fixed to the training input data points, and the single-value precision, which is the same for every kernel, is not an outcome of the learning process and must be thus determined via cross-validation. Several efforts have been done to alleviate this situation, such as Kalman filtering [16], growing and pruning [17], [18], self-organization [19], or covariance update on dense regular node tiling [20], [21]. Despite the previous efforts, and several others [22], not all included here due to space constraints, the adaptation of the Gaussian precision and centers, and the implementation of mechanisms to promote sparsity in the mixture, still remain open problems. As a consequence, current adaptive GFMs [21] cannot beat the "curse of dimensionality."

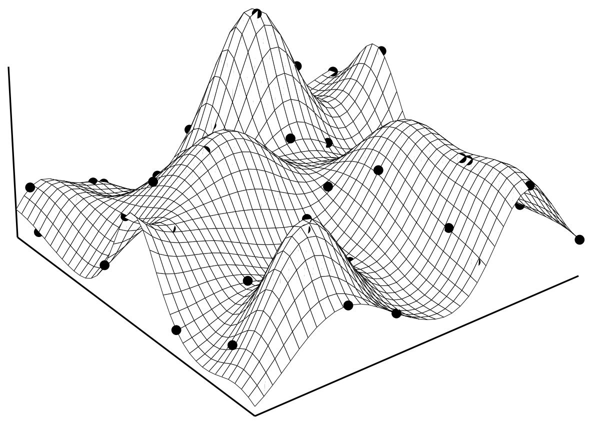
Fig. 1. Sparse multivariate Gaussian function mixture regression. Training data depicted as dots.

Manuscript received Month DD, 2013; revised Month DD, 20WX; accepted Month DD, 20YZ. This paper was recommended by Associate Editor.

L. Weruaga is with Khalifa University of Science, Technology & Research, P.O. Box 573, Sharjah, UAE. Tel: +971 65978834, Fax: +971 65611789, E-mail: weruaga@ieee.org.

J. Vía is with the Department of Communications Engineering, University of Cantabria, 39005 Santander, Cantabria, Spain. Tel: +34 942201493, Fax: +34 942201488, E-mail:jvia@gtas.dicom.unican.es

A preliminary version of this paper, limited to unidimensional non-negative functions, was presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), Vancouver, Canada, May 2013.

Digital Object Identifier

A common trend in the existing late solutions for training mixtures of Gaussian functions [17]–[22] is to use the classical cost criterion built with the error between reference and mixture output, despite it is well known [14], [15] that such a choice corresponds to a high non-linear and non-convex cost function. In this paper we propose a novel optimization criterion to train Gaussian mixtures that seems to overcome the drawbacks and limitations of previous GFM techniques:

- the fitting criterion is based on the generalized logarithmic utility function (GLUF) [23], which can move smoothly between the standard mean square error (MSE) and the logarithmic error minimization,
- it turns out that in this specific scenario, the estimation of all parameters (weight, center, and precision) of all Gaussian functions in the mixture can be achieved by solving a sequence of well-conditioned least squares problems,
- furthermore, sparsity-promoting measures, mainly focused on the precision, can be easily integrated in the original formulation.

The paper is organized as follows: the basic method for nonnegative multivariate functions is presented in Sec. II, Sec. III contains the analysis on the impact of the user-defined parameters such as the GLUF bias and regularization constants, the extension to general (positive/negative) multivariate functions is detailed in Sec. IV, the numerical validation on simulated and real scenarios can be found in Sec. V. Finally, a brief discussion on the computational complexity of the proposed method, as well as the main conclusions of the work, close the paper.

II. Nonnegative Function Identification

The multivariate Gaussian function (1) can be rewritten in a compact fashion as

$$\varphi(\mathbf{x}) = \exp\left(-\left[\mathbf{x}^T\ 1\right]\mathbf{A}\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}\right) \tag{3}$$

where the $(d+1)\times(d+1)$ symmetric matrix $\mathbf{A}$ is defined by

$$\mathbf{A} = \left[\begin{array}{c|c}\mathbf{P} & -\mathbf{P}\mathbf{c}\\ \hline -\mathbf{c}^T\mathbf{P} & \mathbf{c}^T\mathbf{P}\mathbf{c} - \log w\end{array}\right]. \tag{4}$$

The number of distinct parameters in the symmetric matrix $\mathbf{A}$ corresponds to its upper (or lower) triangular matrix

$$N = \frac{(d+1)(d+2)}{2}. \tag{5}$$

Let us start assuming that the function of reference $y(\mathbf{x})$ can be represented as a combination of Gaussians with positive weights.[1] In this paper, we propose a cost function based on the generalized logarithmic utility function (GLUF) [23]

$$u_\sigma(z) = \log(z+\sigma) \tag{6}$$

where $\sigma \geq 0$ is the GLUF bias. In particular, we aim to minimize the mean square error, where the fitting error is defined as

$$e_\sigma(\mathbf{x}) = u_\sigma\big(y(\mathbf{x})\big) - u_\sigma\big(f(\mathbf{x})\big). \tag{7}$$

[1]The more general case in which Gaussians with negative weights are needed is addressed in Section IV.

In this way, the hyper-parameter σ allows us to smoothly move from the traditional case of mean square (linear) error (MSE) minimization ($\sigma \to \infty$) [12],[2] to the minimization of the mean square log error (MSlogE) for $\sigma = 0$, which appears as the "perceptually-linear" scale in speech and audio processing.

It is important to point out that the selection of the order K is a challenging problem. Of course, in some particular applications the parameter K could be *a priori* known. However, we will focus on the general case in which K is unknown, and only an upper bound is available. Thus, we propose to find the optimal number of Gaussians by promoting sparsity in the precision. Hence, the optimization problem to solve contains a regularization term with the "nuclear" norm (or trace norm) of the precision matrix

$$\begin{aligned}&\underset{\mathbf{A}_1,\cdots,\mathbf{A}_K}{\text{minimize}}\ \tfrac{1}{2}\sum_{\mathbf{x}}\big(e_\sigma(\mathbf{x})\big)^2 + \lambda\sum_{j=1}^{K}\mathrm{Tr}(\mathbf{P}_j)\\ &\text{subject to}\ \ \mathbf{P}_1,\cdots,\mathbf{P}_K \succ 0\end{aligned} \tag{8}$$

where $\mathrm{Tr}(\mathbf{B})$ denotes trace of matrix $\mathbf{B}$, and λ is the regularization constant. Excluding the case of mean square log error ($\sigma = 0$) with only one Gaussian shape ($K = 1$, $\lambda = 0$), which can be easily convexified by means of a simple reparameterization [24], the optimization problem in (8) is non-convex in general, and therefore difficult to solve.

Here, we aim to find a local solution by solving the Karush-Kuhn-Tucker (KKT) conditions [25]. In order to do that, we first write the Lagrangian in (8) as

$$\mathcal{L} = \tfrac{1}{2}\sum_{\mathbf{x}}\big(e_\sigma(\mathbf{x})\big)^2 + \sum_{j=1}^{K}\mathrm{Tr}(\mathbf{\Phi}_j\mathbf{A}_j) \tag{9}$$

where

$$\mathbf{\Phi}_j = \left[\begin{array}{c|c}\lambda\mathbf{I} - \mathbf{\Gamma}_j & \mathbf{0}\\ \hline \mathbf{0}^T & 0\end{array}\right]. \tag{10}$$

Here $\mathbf{0}$ is an all-zero vector,[3] $\mathbf{\Gamma}_j$ is the matrix of Lagrange multipliers for the jth Gaussian, and $\mathbf{I}$ is the identity matrix.

A. Gradient and Hessian

In order to devise a numerical method to solve (8), the analysis of gradient and Hessian of the Lagrangian (9) is required. It is simple to deduce that the gradient results in

$$\frac{\partial\mathcal{L}}{\partial\mathbf{A}_k} = \sum_{\mathbf{x}}\rho_k(\mathbf{x})\mathbf{R}_{\mathbf{x}}e_\sigma(\mathbf{x})\ +\ \mathbf{\Phi}_k \tag{11}$$

where matrix $\mathbf{R}_{\mathbf{x}}$ is the autocorrelation matrix of point $\mathbf{x}$

$$\mathbf{R}_{\mathbf{x}} = \begin{bmatrix}\mathbf{x}\\1\end{bmatrix}\left[\mathbf{x}^T\ 1\right] \tag{12}$$

and $\rho_k(\mathbf{x})$ is the "relevance" of the kth Gaussian at point $\mathbf{x}$

$$\rho_k(\mathbf{x}) = \frac{\varphi_k(\mathbf{x})}{f(\mathbf{x})+\sigma} \tag{13}$$

[2]Note that $u_\sigma(a) - u_\sigma(b) \approx \frac{1}{\sigma}(a-b)$ for $\sigma \gg a, b$.

[3]The length of all-one $\mathbf{1}$ or all-zero $\mathbf{0}$ column vector (or matrix) is not shown in the notation as it can be deduced from the context.

which satisfies $0 < \rho_k(\mathbf{x}) < 1$. The error (7) can be written around the kth function as follows

$$\begin{aligned} e_\sigma(\mathbf{x}) &= \log\big(y(\mathbf{x}) + \sigma\big) + \log \rho_k(\mathbf{x}) - \log \varphi_k(\mathbf{x}) \\ &= y_k(\mathbf{x}) + \begin{bmatrix}\mathbf{x}^T & 1\end{bmatrix} \mathbf{A}_k \begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix} \end{aligned} \tag{14}$$

where the reference $y_k(\mathbf{x})$ is

$$y_k(\mathbf{x}) = \log\big(y(\mathbf{x}) + \sigma\big) + \log \rho_k(\mathbf{x}). \tag{15}$$

Note that in the gradient (11), the error at each sample $\mathbf{x}$ is weighted by the relevance function $\rho_k(\mathbf{x})$.

The Hessian matrix allows us to understand the nature of the Lagrangian (hyper)surface. The Hessian is composed of $N \times N$ submatrices, each one relating two Gaussian functions. In case of different functions, the submatrix corresponds to

$$\begin{aligned} \frac{\partial^2 \mathcal{L}}{\partial \mathbf{A}_k \partial \mathbf{A}_\ell} &= \sum_{\mathbf{x}} \rho_k(\mathbf{x}) \rho_\ell(\mathbf{x}) \left(\mathbf{R}_{\mathbf{x}} \otimes \mathbf{R}_{\mathbf{x}}\right) \\ &\quad + \sum_{\mathbf{x}} \rho_k(\mathbf{x}) \rho_\ell(\mathbf{x})\, e_\sigma(\mathbf{x}) \left(\mathbf{R}_{\mathbf{x}} \otimes \mathbf{R}_{\mathbf{x}}\right) \end{aligned} \tag{16}$$

where $\otimes$ denotes Kronecker product.[4] For the same function, the Hessian results in

$$\begin{aligned} \frac{\partial^2 \mathcal{L}}{\partial \mathbf{A}_k^2} &= \sum_{\mathbf{x}} \rho_k^2(\mathbf{x}) \left(\mathbf{R}_{\mathbf{x}} \otimes \mathbf{R}_{\mathbf{x}}\right) \\ &\quad - \sum_{\mathbf{x}} \rho_k(\mathbf{x}) \big(1 - \rho_k(\mathbf{x})\big) e_\sigma(\mathbf{x}) \left(\mathbf{R}_{\mathbf{x}} \otimes \mathbf{R}_{\mathbf{x}}\right). \end{aligned} \tag{17}$$

The Hessian (16)-(17) presents essentially two terms, the first one being positive semidefinite, while the second one, that cannot be guaranteed such, is proportional to the error.

B. Numerical Algorithm

The KKT conditions can be written as

$$\begin{array}{ll} \text{Gradient of the Lagrangian:} & \dfrac{\partial \mathcal{L}}{\partial \mathbf{A}_k} = 0 \quad \forall k \\ \text{Primal feasibility:} & \mathbf{P}_k \succ 0 \\ \text{Dual feasibility:} & \mathbf{\Gamma}_k \succ 0 \\ \text{Complementary Slackness:} & \mathrm{Tr}(\mathbf{\Gamma}_k \mathbf{P}_k) = 0. \end{array} \tag{18}$$

For the sake of simplicity in the presentation of the numerical method, we reorder the Gaussian parameter matrix $\mathbf{A}_k$ in a column vector $\mathbf{a}_k$; in a similar way, the autocorrelation matrix $\mathbf{R}_{\mathbf{x}}$ can be "vectorized" in the column vector $\mathbf{r}_{\mathbf{x}}$.[5] According to this notation, the error (14) can be simply written as

$$e_\sigma(\mathbf{x}) = y_k(\mathbf{x}) + \mathbf{r}_{\mathbf{x}}^T \mathbf{a}_k. \tag{19}$$

The solution equation resulting from the null of the gradient (11) can be also written as

$$\sum_{\mathbf{x}} \rho_k(\mathbf{x})\, \mathbf{r}_{\mathbf{x}}\, e_\sigma(\mathbf{x}) + \boldsymbol{\phi}_k = \mathbf{0} \tag{20}$$

where $\boldsymbol{\phi}_k$ corresponds to the vectorization of matrix $\mathbf{\Phi}_k$ (10). As the relevance $\rho_k(\mathbf{x})$ depends on all Gaussian parameters to be found, the null of the gradient results in a complicated system of nonlinear equations.

Rather than solving (20), we propose to replace it with a solution equation containing a closer approximation of the Hessian matrix: note that in the positive-(semi)definite term of the Hessian (17), the importance of each point $\mathbf{x}$ is weighted by the *square* of its relevance; hence, we redefine the solution equation (20) accordingly as

$$\sum_{\mathbf{x}} \rho_k^2(\mathbf{x})\, \mathbf{r}_{\mathbf{x}}\, e_\sigma(\mathbf{x}) + \boldsymbol{\phi}_k = \mathbf{0}. \tag{21}$$

Note that if a solution to (20) results in perfect fitting, i.e. $e_\sigma(\mathbf{x}) = 0$, it is also a solution to (21).

Upon replacement of the error (19), equation (21) becomes

$$\sum_{\mathbf{x}} \rho_k^2(\mathbf{x})\, \mathbf{r}_{\mathbf{x}}\, y_k(\mathbf{x}) + \sum_{\mathbf{x}} \rho_k^2(\mathbf{x})\, \mathbf{r}_{\mathbf{x}} \mathbf{r}_{\mathbf{x}}^T\, \mathbf{a}_k = -\boldsymbol{\phi}_k \tag{22}$$

which can be summarized into

$$\mathbf{g}_k + \mathbf{H}_k \mathbf{a}_k = \begin{bmatrix} \boldsymbol{\gamma}_k \\ \mathbf{0} \end{bmatrix} \tag{23}$$

where

$$\mathbf{H}_k = \sum_{\mathbf{x}} \rho_k^2(\mathbf{x}) \left(\mathbf{r}_{\mathbf{x}} \mathbf{r}_{\mathbf{x}}^T\right) \tag{24a}$$

$$\mathbf{g}_k = \sum_{\mathbf{x}} \rho_k^2(\mathbf{x})\, y_k(\mathbf{x})\, \mathbf{r}_{\mathbf{x}} + \lambda \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} \tag{24b}$$

and vector $\boldsymbol{\gamma}_k$ corresponds to the vectorization of the Lagrange multipliers matrix $\mathbf{\Gamma}_k$. The solution to (23) must be accomplished in an iterative fashion by considering the relevance $\rho_k(\mathbf{x})$ to be fixed (hence so the terms $\mathbf{H}_k$ and $\mathbf{g}_k$) and reevaluating the system terms (24) with the resulting partial solution. Such an approach corresponds to the following quasi-Newton recursion

$$\mathbf{a}_k^{(\xi+1)} = -\left(\mathbf{H}_k^{(\xi)}\right)^{\dagger} \mathbf{g}_k^{(\xi)} \tag{25}$$

where ξ denotes iteration and $\dagger$ pseudo inverse. Note that the system matrix (24a) is equal to the positive-(semi)definite term in the Hessian (17). This fact has a positive impact in the speed of convergence of the algorithm (25), in special when the convergence is approaching the (local) minimum.

So far we have optimistically assumed that the constraints in the original optimization problem (8) are satisfied, case for which the Lagrange multipliers $\boldsymbol{\gamma}_k$ in (23) vanish. At each iteration (25), we need however to check if the positive definiteness constraint is active in the resulting partial solution, which implies $\mathbf{P}_k^{(\xi+1)} \not\succ 0$ (equivalently $\boldsymbol{\gamma}_k \neq \mathbf{0}$); if so,[6] the resulting precision matrix is not valid and it must be thus replaced with a (appropriate) positive definite matrix.

[4] Let $\mathbf{B}$ be an $m \times n$ matrix and $\mathbf{C}$ a $p \times q$ matrix, then the Kronecker product $\mathbf{B} \otimes \mathbf{C}$ is the $mp \times nq$ block matrix

$$\mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} b_{11}\mathbf{C} & \cdots & b_{1n}\mathbf{C} \\ \vdots & \ddots & \vdots \\ b_{m1}\mathbf{C} & \cdots & b_{mn}\mathbf{C} \end{bmatrix}.$$

[5] Needless to say that any reordering is valid. However, for the sake of consistency, we assume that vector $\mathbf{a}$ results from vectorizing matrix $\mathbf{A}$ (4) as follows: $\mathbf{a} = [\mathbf{p}_0; \mathbf{p}_1; \mathbf{p}_{-1}; \cdots; \mathbf{p}_d; \mathbf{p}_{-d}; \mathbf{A}_{(1:d,d+1)}; \mathbf{A}_{(d+1,:)}]$, where $\mathbf{p}_j$ is the jth diagonal of matrix $\mathbf{P}$, and $\mathbf{A}_{(:,d+1)}$ is $\mathbf{A}$'s rightmost column.

[6] There exist cost-effective methods to check the positive definiteness of a matrix without recurring to an expensive singular value decomposition.

- An option is to project the resulting precision $\mathbf{P}_k = \mathbf{\Theta}_k \mathbf{\Lambda}_k \mathbf{\Theta}_k^T$ (where $\mathbf{\Theta}_k$ is a unitary rotation matrix and $\mathbf{\Lambda}_k$ is a diagonal matrix) onto the space of positive definite matrices, an operation accomplished with $\mathbf{Q}_k = \mathbf{\Theta}_k \lfloor \mathbf{\Lambda}_k \rfloor_0 \mathbf{\Theta}_k^T$, where $\lfloor \ \rfloor_0$ denotes lower clipping the matrix elements by 0.
- The main question now is whether the tentative precision matrix $\mathbf{Q}_k$ yields a reduction in the cost, that is, if $\mathcal{L}(\mathbf{Q}_k) \le \mathcal{L}(\mathbf{P}_k)$; because the Lagrangian is not convex, such a projection needs not result in a cost decrease (it may in fact yield the opposite undesired result).
- A simple and safe alternative is simply to keep the original positive-definite precision matrix as it does not yield a cost increase; in fact, estimating the remaining $d+1$ parameters (related to the center and the amplitude of the Gaussian) promotes a reduction of the cost.

In consequence, the number of parameters left to estimate reduces to just $d+1$, which correspond to $\mathbf{A}_k$'s rightmost column (4); let $\mathbf{D}$ be the diagonal matrix filled with 1 only in the positions corresponding to the remaining $d+1$ parameters, the problem left to solve is

$$\mathbf{g}_k^{(\xi)} + \mathbf{H}_k^{(\xi)}\big((\mathbf{I}-\mathbf{D})\mathbf{a}_k^{(\xi)} + \mathbf{D}\mathbf{b}_k\big) = \mathbf{0} \tag{26}$$

which is accomplished by[7]

$$\mathbf{b}_k = -\Big(\mathbf{H}_k^{(\xi)}\mathbf{D}\Big)^{\dagger}\big(\mathbf{g}_k^{(\xi)} + \mathbf{H}_k^{(\xi)}(\mathbf{I}-\mathbf{D})\mathbf{a}_k^{(\xi)}\big). \tag{27}$$

The final Gaussian parameters are obtained by joining the initial precision matrix and the resulting $d+1$ parameters (27)

$$\mathbf{a}_k^{(\xi+1)} = (\mathbf{I}-\mathbf{D})\mathbf{a}_k^{(\xi)} + \mathbf{D}\mathbf{b}_k. \tag{28}$$

C. Problem Condition

Parametric data fitting is by nature an ill-posed problem, in especial when the number of parameters to estimate is larger than the size of the training data. Here the appropriate estimation of the kth Gaussian parameters can be compromised when matrix $\mathbf{H}_k$ becomes ill conditioned.[8] In such an event, avoiding the inverse problem at all may be even preferable. A less conservative option though is to introduce a regularization with respect to a reference point, that is, to solve

$$\begin{aligned} \underset{\mathbf{A}_1^{(\xi)},\cdots,\mathbf{A}_K^{(\xi)}}{\text{minimize}} \quad & \mathcal{L}^{(\xi)} + \mu \sum_{j=1}^{K} \big\|\mathbf{\Delta}_j^{(\xi)}\big\|^2 \\ \text{subject to} \quad & \mathbf{P}_1^{(\xi)}, \cdots, \mathbf{P}_K^{(\xi)} \succ 0 \end{aligned} \tag{29}$$

where $\mathbf{\Delta}_j^{(\xi)} = \mathbf{A}_j^{(\xi)} - \mathbf{A}_j^{(\xi-1)}$, $\mu > 0$ is the regularization hyper-parameter. The regularization term is aimed to prevent consecutive solutions from differing by a large amount. It is simple to deduce that, with this modification (29), the quasi-Newton iteration (25) becomes

$$\mathbf{a}_k^{(\xi+1)} = -\Big(\mathbf{H}_k^{(\xi)} + \mu\mathbf{I}\Big)^{-1}\big(\mathbf{g}_k^{(\xi)} - \mu\mathbf{a}_k^{(\xi)}\big). \tag{30}$$

[7]Note that the operation (27) is accomplished by the inversion of a $(d+1)$-column square matrix.

[8]Matrix $\mathbf{H}_k$ corresponds to a N-point local linear fitting problem, for which at least N different points are required for a unique solution.

The resulting diagonal loading of the system matrix $\mathbf{H}_k$ is well known to alleviate the possible ill-condition of that matrix. It is also well known that such a regularization may come at the expense of slowing down the convergence of the algorithm. Variable loading alternatives [26] with a good compromise between regularization and speed could be adopted in this scenario though. This option can be stated with the following optimization problem

$$\begin{aligned} \underset{\mathbf{A}_1^{(\xi)},\cdots,\mathbf{A}_K^{(\xi)}}{\text{minimize}} \quad & \mathcal{L}^{(\xi)} + \mu \sum_{j=1}^{K} \mathbf{\Delta}_j^{T(\xi)}\mathbf{T}_j^{(\xi-1)}\mathbf{\Delta}_j^{(\xi)} \\ \text{subject to} \quad & \mathbf{P}_1^{(\xi)}, \cdots, \mathbf{P}_K^{(\xi)} \succ 0 \end{aligned} \tag{31}$$

where $\mathbf{T}_k^{(\xi)}$ is the positive definite regularization matrix, which may be parametric and will thus have to be evaluated at every iteration. For instance, in (29) this matrix simply corresponds to the identity, $\mathbf{T}_k = \mathbf{I}$, but in [26] it is built with the inverse of the system matrix, $\mathbf{T}_k = \mathbf{H}_k^{-1}$. Further study on such alternatives is beyond the scope of the paper but worth undertaking.

D. Algorithm Initialization

Not less important is the way the resulting iterative algorithm is initialized. Obviously, the Gaussian functions must be initially placed in the proximity of the data. Formally, one needs to prevent $\rho_k(\mathbf{x}) \simeq 0$, $\forall k$ for any datapoint $\mathbf{x}$, which would imply that the point $\mathbf{x}$ is being "ignored" by all Gaussian functions. In addition to that, the functions must sufficiently overlap with each other. These requirements point out to a method for clustering high-dimensional data as initialization mechanism. The first solution we considered was the popular Gaussian mixture models (GMM) trained with the EM algorithm [27]. This method results in an heterogeneous Gaussian mixture that represents the input data under a probabilistic perspective. However, given that the input data $\mathbf{x}$ simply corresponds to locations at which the reference function has been sampled, it results thus more appropriate to cluster the data on regions with similar shape and size. An adequate method for that purpose is the popular k-means algorithm. This method guarantees a uniform clustering with Voronoi regions of similar (hyper)volume. This part covers the initialization of center and precision; the weight initialization is not critical, simply $w > \sigma$.

E. Alternative Model

Keeping all Gaussian parameters under a unique matrix $\mathbf{A}$ (4) results in a compact formulation and a simple system matrix (24). However, one might like to be able to constraint the Gaussian center $\mathbf{c}$ to a certain fixed position (to the input data points for instance) or within a certain area, or to include sparsity measures over the Gaussian weights w. Setting constraints to the center and/or the weight is however problematic because those terms are somewhat hidden in the parametric matrix $\mathbf{A}$.

An alternative parameterization to address these intentions is namely the original one (1), which we bring here again

Algorithm 1 Sparse Multivariate Gaussian Regression

```
input: $\mathbf{x}$, $y(\mathbf{x})$ and $\sigma$, $\lambda$, $\mu$, $K$.
output: Gaussian parameters $\mathbf{Z}_k$ (or in vector form $\mathbf{z}_k$).
initialize $\mathbf{Z}_k^{(1)}$ for $k = 1, \cdots, K$, such that $\mathbf{P}_k^{(1)} \succ 0$
repeat
  $f^{(\xi)}(\mathbf{x}) = \sum_{k=1}^{K} \varphi_k^{(\xi)}(\mathbf{x})$
  $e_\sigma^{(\xi)}(\mathbf{x}) = \log\big(y(\mathbf{x}) + \sigma\big) - \log\big(f^{(\xi)}(\mathbf{x}) + \sigma\big)$
  for $k$th Gaussian function do
    $\rho_k^{(\xi)}(\mathbf{x}) = \varphi_k^{(\xi)}(\mathbf{x}) / \big(f^{(\xi)}(\mathbf{x}) + \sigma\big)$.
    $\mathbf{e}_k^{(\xi)} = \sum_{\mathbf{x}} \big(\rho_k^{(\xi)}(\mathbf{x})\big)^2 \mathbf{s}_{\mathbf{x}k}^{(\xi)} e_\sigma^{(\xi)}(\mathbf{x}) + \lambda\big[\mathbf{1}^T \ \mathbf{0}^T\big]^T$
    $\mathbf{H}_k^{(\xi)} = \sum_{\mathbf{x}} \big(\rho_k^{(\xi)}(\mathbf{x})\big)^2 \mathbf{s}_{\mathbf{x}k}^{(\xi)} \mathbf{s}_{\mathbf{x}k}^{(\xi)T}$
    $\mathbf{z}_k^{(\xi+1)} = \mathbf{z}_k^{(\xi)} + \Big(\mathbf{H}_k^{(\xi)} + \mu\mathbf{I}\Big)^{-1} \mathbf{e}_k^{(\xi)}$
    if $\mathbf{P}_k^{(\xi+1)} \not\succ 0$ then
      $\mathbf{z}_k^{(\xi+1)} = \mathbf{z}_k^{(\xi)} + \Big(\mathbf{D}\big(\mathbf{H}_k^{(\xi)} + \mu\mathbf{I}\big)\mathbf{D}\Big)^{\dagger} \mathbf{e}_k^{(\xi)}$
    end if
  end for
until convergence
```

albeit in a more compact form

$$\varphi(\mathbf{x}) = \exp\big(-(\mathbf{x}-\mathbf{c})^T\mathbf{P}(\mathbf{x}-\mathbf{c}) + \log w\big). \tag{32}$$

Precision $\mathbf{P}$, center $\mathbf{c}$ and weight are now explicit parameters in the model. We reorder the Gaussian parameters in a single matrix as follows

$$\mathbf{Z} = \left[\begin{array}{c|c} \mathbf{P} & \mathbf{c} \\ \hline \mathbf{c}^T & \log w \end{array}\right]. \tag{33}$$

The gradient of the Lagrangian results in this case in

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}_k} = \sum_{\mathbf{x}} \rho_k(\mathbf{x})\, \mathbf{S}_{\mathbf{x}k}\, e_\sigma(\mathbf{x}) + \mathbf{\Phi}_k \tag{34}$$

where

$$\mathbf{S}_{\mathbf{x}k} = \left[\begin{array}{c|c} (\mathbf{x}-\mathbf{c}_k)(\mathbf{x}-\mathbf{c}_k)^T & -\mathbf{P}_k(\mathbf{x}-\mathbf{c}_k) \\ \hline -(\mathbf{x}-\mathbf{c}_k)^T\mathbf{P}_k & -1 \end{array}\right]. \tag{35}$$

On the other hand, the Hessian presents the following three terms

$$\begin{aligned}\frac{\partial^2 \mathcal{L}}{\partial \mathbf{Z}_k^2} &= \sum_{\mathbf{x}} \rho_k^2(\mathbf{x})\big(\mathbf{S}_{\mathbf{x}k} \otimes \mathbf{S}_{\mathbf{x}k}\big) \\ &+ \sum_{\mathbf{x}} \rho_k(\mathbf{x})\big(1-\rho_k(\mathbf{x})\big) e_\sigma(\mathbf{x}) \big(\mathbf{S}_{\mathbf{x}k} \otimes \mathbf{S}_{\mathbf{x}k}\big) \\ &+ \sum_{\mathbf{x}} \rho_k(\mathbf{x})\, e_\sigma(\mathbf{x}) \frac{\partial \mathbf{S}_{\mathbf{x}k}}{\partial \mathbf{Z}_k}. \end{aligned} \tag{36}$$

Only the first term in the Hessian (36) is positive (semi)definite. This term presents a polynomial dependance with the Gaussian parameters, hence the associated Lagrangian surface is not quadratic (as it was in case of (17)), but it is convex. Thus, a quasi-Newton algorithm, similar to the one disclosed in Sec. II-B, is appropriate to accomplish the solution. Such a numerical method is brought in Algorithm 1.

With this alternative parametric model (32), the amplitude w of the Gaussian function is embedded in a system parameter. Thus, additional constraints can be imposed to the Gaussian weights $\boldsymbol{w} = [w_1, \cdots, w_K]$. For instance, a popular methodology to achieve sparsity on a weighted mixture is

$$\underset{\mathbf{Z}_k}{\text{minimize}} \quad \sum_{\mathbf{x}} \big(e_\sigma(\mathbf{x})\big)^2 + \delta \|\boldsymbol{w}\|_p \tag{37}$$

where the additional regularization term $\|\boldsymbol{w}\|_p$ corresponds to the p-norm ℓ_p (in the weight space)

$$\|\boldsymbol{w}\|_p = \sum_{k=1}^{K} |w_k|^p \tag{38}$$

and the hyper-parameter $\delta \geq 0$ is the regularization tradeoff. It is now well understood that any norm such $0 < p \leq 1$ corresponds to natural mathematical measures of sparsity [28] (unlike ℓ_2). The case ℓ_1 captures special interest because it does not compromise the convexity of the main cost function. However, since the regularization term (38) is built with the p-power of the amplitude, while the parameter to estimate corresponds to its logarithm, this problem (37) is not easy to solve numerically in a compact fashion. A tentative methodology to include this regularization is namely to alternate the original method with the ℓ_p weight penalization algorithm. In case of ℓ_1, a reweighed LS weight update would be

$$w_k^{(\xi+1)} = w_k^{(\xi)} \cdot \frac{w_k^{(\xi)}}{w_k^{(\xi)} + \delta}. \tag{39}$$

On the other hand, constraints on the center (and obviously on the precision) can be also easily embedded in the optimization problem. For instance, setting the centers to a fixed position simply results in a reduction (by d) of the system parameters, while keeping the centers within certain margins can be accomplished with a similar methodology to the one detailed around (26).

III. Impact of User-Defined Constants

A. Mean Square Log-Error (MSlogE) Criterion

The GLUF bias σ allows us to move from the MSlogE ($\sigma = 0$) to the classical MSE minimization ($\sigma \to \infty$). This section covers the first case. For $\sigma = 0$ the relevance function $\rho_k(\mathbf{x})$ takes a value close to 0 in those regions where the kth Gaussian function is not dominant, while in neighbouring regions to the Gaussian the relevance gets often close to 1. This fact makes the non-convex term in the Hessian (17) negligible because $\rho(1-\rho) \simeq 0$ at the extremes $\rho \simeq 0$ and $\rho \lesssim 1$. In summary, the resulting Hessian (17) depends only on the input data $\mathbf{x}$ in a vicinity of the Gaussian function, and the cost function can be thus considered "locally" quadratic. The importance of the relevance function is illustrated with a toy unidimensional experiment in Fig. 2: each Gaussian function, which in logarithmic scale results as an inverted parabola, follows closely its reference (15) over the local region determined by the respective relevance function.

In regard to the Hessian cross terms (16), and supported on the previous reasoning, we can state that the product between the relevance of weakly-overlapping Gaussian functions becomes negligible, that is, $\rho_k\rho_\ell \simeq 0$ for $k \neq \ell$. The independent

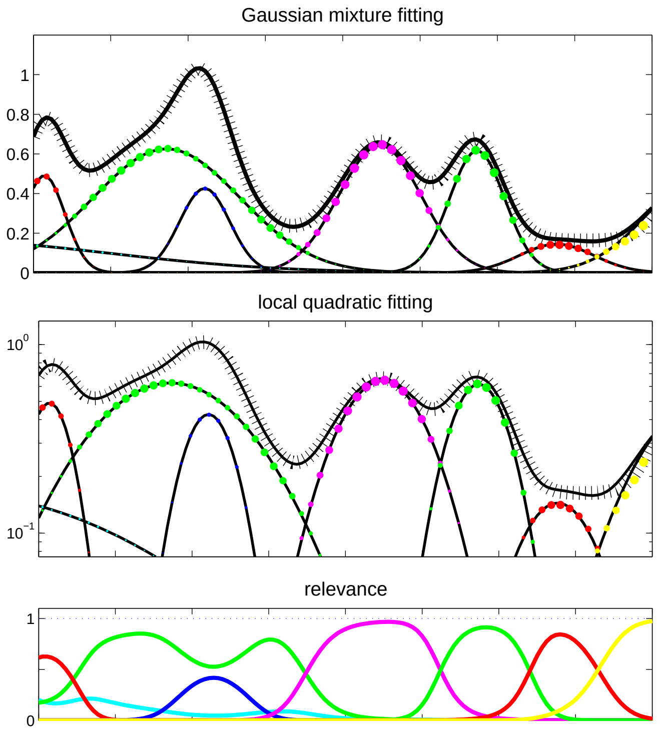

Fig. 2. Gaussian mixture fitting for GLUF bias $\sigma = 0$. Gaussian mixture in linear scale (top), in logarithmic scale (middle), and corresponding relevance functions (bottom). Dots correspond to the references (15) and their size reflects the importance of the data in the estimation.

update of each Gaussian function, as proposed with (20), is thus an adequate and computationally efficient approach (versus inverting the global Hessian).

B. Mean Square Error (MSE) Minimization

The classical MSE minimization presents manifesting differences with respect to the previous case. When the GLUF bias is very large, $\sigma \to \infty$, the Hessian (36) (scaled by σ^2) converges to

$$\lim_{\sigma\to\infty} \sigma^2 \frac{\partial^2 \mathcal{L}}{\partial \mathbf{Z}_k^2} = \sum_{\mathbf{x}} \varphi_k^2(\mathbf{x}) \big(\mathbf{S}_{\mathbf{x}k} \otimes \mathbf{S}_{\mathbf{x}k}\big) + \sum_{\mathbf{x}} \varphi_k(\mathbf{x}) e(\mathbf{x}) \left(\mathbf{S}_{\mathbf{x}k} \otimes \mathbf{S}_{\mathbf{x}k} + \frac{\partial \mathbf{S}_{\mathbf{x}k}}{\partial \mathbf{Z}_k} \right) \quad (40)$$

where $e(\mathbf{x})$ corresponds to the linear error

$$e(\mathbf{x}) = y(\mathbf{x}) - f(\mathbf{x}) \quad (41)$$

because $\rho(1-\rho) \to \rho$, $\sigma\rho \to \varphi$ and $\sigma e_\sigma \to e$. On the other hand, the full gradient (scaled by σ^2 as well) becomes

$$\lim_{\sigma\to\infty} \sigma^2 \frac{\partial \mathcal{L}}{\partial \mathbf{Z}_k} = \sum_{\mathbf{x}} \varphi_k(\mathbf{x})\, \mathbf{S}_{\mathbf{x}k}\, e(\mathbf{x}). \quad (42)$$

This result (42) matches, as expected, the expression already deduced for the generalized Gaussian radial basis functions (RBF) [14]. It is worth mentioning that this former MSE-driven RBF training is based on a plain gradient-descent algorithm, where the value of the step sizes (for w, $\mathbf{c}$ and $\mathbf{P}$) must be properly selected to assure the algorithm convergence. Given the difficulty for maintaining stability in the precision, in the practice only weights and centers are updated.

The MSE criterion, under the GLUF-based analysis in (40) and (42), presents the following two distinctive characteristics:

1) The error at each point $\mathbf{x}$ turns out weighted by the very activation function $\varphi_k(\mathbf{x})$. As the relevance function $\rho_k(\mathbf{x})$ is thus present in neither the Hessian (40) nor the gradient (42), competition among the Gaussian functions for the data does not exist.
2) The error (41) has a severe non-linear dependence with the Gaussian parameters, and it cannot be thus reordered in a nearly-linear relation as (14).

From our experience, the performance of the proposed method for large σ values, such as $\sigma > y(\mathbf{x})$, results in slow and stiff convergence. Loosely speaking, as competition among the Gaussian functions here does not exist, each one deals with the data in its vicinity, determined by $\varphi_k(\mathbf{x})$, regardless of which other Gaussian functions are there. This way of independent working, instead of cooperative, leads to poor suboptimal solutions. We intuit that the lack of relevance $\rho_k(\mathbf{x})$ here could be replaced by a global Hessian involving all parameters from all Gaussian functions. That option is however difficult to implement and computationally prohibitive. In consequence our attempts at devising a Newton (or quasi-Newton) algorithm based on the Hessian (40) and the gradient (42) has not led to conclusive results. Therefore, if an MSE minimization is strictly required, we suggest to train the system for low/moderate values of σ, and upon convergence, to train only the weights $\boldsymbol{w}$ under the MSE criterion.

C. Sparsity Control: λ and δ

Pursuing sparsity on the precision (8) may somewhat compromise the fitting task itself, that is, by forcing the precision to be low, the fitting performance (or MSE) may falter. This fact raises the question about the best value of the regularization constant λ for a given problem: a big value yields "wide" overlapping Gaussians functions, unable to fit gently in the data; on the other hand, a low value may not promote enough a sparse mixture, thus likely to produce an over-fitted solution. As the optimal regularization hyper-parameter depends on the structure of the very data [29], its analytical determination is in the practice difficult. Supported on this discussion, the iterative training algorithm is suggested with two sequential stages (similar strategy is used in compressive sensing):

1) in the first stage, the regularization on the precision is active, that is, $\lambda > 0$; the MSE at every iteration is evaluated and monitored: when the MSE experiences a steady increase/stall, the second stage becomes active,
2) in the second stage, the regularization hyper-parameter is set to zero, $\lambda = 0$ (or decreased monotonically), until final convergence.

A desired outcome of enforcing sparsity in the precision by means of $\lambda > 0$ is that of neighbouring Gaussian functions converging to an equivalent function if they are not useful in reducing the MSE. This function merge applies to the center and precision, but not to the weight. This situation, although effectively sparse, is not such from a computational perspective as non-zero weight elements are required to evaluate the Gaussian mixture. The inclusion of sparsity measures in the weight (39) by means of δ helps get rid of those undesired replicas, thus achieving explicit mixture sparsity.

IV. Extension to General Functions

The condition $y(\mathbf{x}) \geq 0$ renders the proposed technique inadequate with general functions (that is positive and/or negative) because the original error (7) may include the logarithm of negative numbers. A straightforward (and naïve) way to extend the previous methodology to general functions is to consider the new nonnegative function of reference $b + y(\mathbf{x})$ such that $b > |y(\mathbf{x})|$ instead. The resulting solution consists thus of a nonnegative Gaussian mixture plus a negative constant term equal to $-b$ (which can be treated as a Gaussian of null precision). One could also choose the nonnegative function $b - y(\mathbf{x})$ as reference function instead, which would yield a positive constant term equal to b plus a negative Gaussian mixture, a solution that differs clearly from the first alternative. These cases unveil the presence of multiple (possibly infinite) valid solutions.

We opt to follow a different path by defining the fitting function as follows

$$f(\mathbf{x}) = f^{+}(\mathbf{x}) - f^{-}(\mathbf{x}) \tag{43}$$

where $f^{+}(\mathbf{x}) \geq 0$ is the positive contribution of the mixture, and $f^{-}(\mathbf{x}) \geq 0$ the negative one. In relation to those, we introduce two references $y^{+}(\mathbf{x})$ and $y^{-}(\mathbf{x})$ defined as

$$y^{+}(\mathbf{x}) = \quad y(\mathbf{x}) + f^{-}(\mathbf{x}) \tag{44a}$$
$$y^{-}(\mathbf{x}) = -y(\mathbf{x}) + f^{+}(\mathbf{x})\,. \tag{44b}$$

Let us assume for a moment that $f^{-}(\mathbf{x})$ were a priori available, hence $y^{+}(\mathbf{x})$ would be a proper reference for estimating $f^{+}(\mathbf{x})$; obviously $f^{-}(\mathbf{x})$ is unknown, but the proposed statement (44) unveils the fact that the estimation of $f^{+}(\mathbf{x})$ is reciprocally linked to the estimation of $f^{-}(\mathbf{x})$.

We thus rephrase the problem as that of obtaining the following nonnegative function

$$F(\mathbf{x}) = f^{+}(\mathbf{x}) + f^{-}(\mathbf{x}) \tag{45}$$

and define the fitting error as

$$e_\sigma(\mathbf{x}) = u_\sigma\big(y^{+}(\mathbf{x}) + y^{-}(\mathbf{x})\big) \;-\; u_\sigma\big(F(\mathbf{x})\big) \tag{46}$$

enforcing the obvious constraints $y^{+}(\mathbf{x}) \geq 0$ and $y^{-}(\mathbf{x}) \geq 0$. The optimization problem to solve becomes thus

$$\begin{aligned} \underset{f^{+},f^{-}}{\text{minimize}} \quad & \sum_{\mathbf{x}} \big(e_\sigma(\mathbf{x})\big)^2 + \lambda \sum_{j=1}^{K} \mathrm{Tr}(\mathbf{P}_j) + \delta \|\boldsymbol{w}\|_1 \\ \text{subject to} \quad & y^{+}(\mathbf{x}), y^{-}(\mathbf{x}) \geq 0 \;\text{ and }\; \mathbf{P}_1, \cdots, \mathbf{P}_K \succ 0. \end{aligned} \tag{47}$$

The weight-regularization term $\|\boldsymbol{w}\|_1$ is present because the trivial case $y(\mathbf{x}) = 0$ yields $f^{+}(\mathbf{x}) = f^{-}(\mathbf{x})$, which has infinite solutions; we are thus only interested in the solution $f^{+}(\mathbf{x})$, $f^{-}(\mathbf{x}) = 0$, which is promoted by the additional regularization term.

In what follows, we outline the strategy to solve this optimization problem, focusing on the positive contribution f^{+} of the mixture.

- The condition $y^{+}(\mathbf{x}) \geq 0$ can be enforced by lower zero-clipping the reference itself.
- Regarding condition $y^{-}(\mathbf{x}) \geq 0$, at any iteration the best (non-negative) estimation of $y^{-}(\mathbf{x})$ is the very negative part of the mixture $f^{-}(\mathbf{x})$.

By following the previous arguments, we redefine the global error (46) for the positive term as follows

$$\begin{aligned} e^{+}_\sigma(\mathbf{x}) &= u_\sigma\big(\lfloor y^{+}(\mathbf{x})\rfloor_0 + f^{-}(\mathbf{x})\big) \;-\; u_\sigma\big(F(\mathbf{x})\big) \\ &= u_{\sigma+f^{-}(\mathbf{x})}\big(\lfloor y^{+}(\mathbf{x})\rfloor_0\big) \;-\; u_{\sigma+f^{-}(\mathbf{x})}\big(f^{+}(\mathbf{x})\big) \end{aligned} \tag{48}$$

where $\lfloor z \rfloor_\epsilon$ denotes lower clipping z by ϵ. Based on the alternative parametric model detailed in Sec. II-E, the solution equation here results in

$$\sum_{\mathbf{x}} \rho^2_k(\mathbf{x})\, \mathbf{S}_{\mathbf{x}k}\, e^{+}_\sigma(\mathbf{x}) \;+\; \mathbf{\Phi}_k = \mathbf{0} \tag{49}$$

for $k \in f^{+}$, where the relevance function is

$$\rho_k(\mathbf{x}) = \frac{\varphi_k(\mathbf{x})}{f^{+}(\mathbf{x}) + f^{-}(\mathbf{x}) + \sigma}\,. \tag{50}$$

The solution equation for the negative contribution f^{-} of the mixture is obtained by simply swapping superscript $^{+}$ and $^{-}$ in the former equations. The numerical algorithm to solve this problem is composed of two numerical processes, following exactly the original one detailed in Sec. II-B, each one for the update of the positive and for the negative contributions in the mixture. These two numerical processes work actually as one, as the error (48) and the relevance (50) are both built with all Gaussian functions.

V. Results

In order to illustrate the qualitative performance of the proposed method we use several synthetic scenarios with low dimensions. A quantitative performance analysis is also carried out on realistic scenarios of large dimensional data.

A. Synthetic Scenarios

The aim of the first experiment is to demonstrate the effect of the regularization for the precision and that for the weight, managed by the hyper-parameters λ and δ respectively, on a simple yet insightful scenario. The training dataset consists of $L = 40$ points in the interval $[0, 6]$ sampling the chirp function

$$y(x) = \cos\big(2x + 0.08x^3\big).$$

The general method in Sec. IV was used, the size of the Gaussian mixture was $K = 40$, and the GLUF bias is small $\sigma = 0.01$, and the diagonal loading was set to $\mu = 0.1$. Three methods were used in the experiment:

1) No regularization on the precision, $\lambda = 0$, but regularization on the weights $\delta = 0.05$.
2) Regularization on the precision, $\lambda = 0.001$ (as well as on the weights $\delta = 0.05$).
3) Least-squares support vector regression (LS-SVR) with Gaussian kernel.

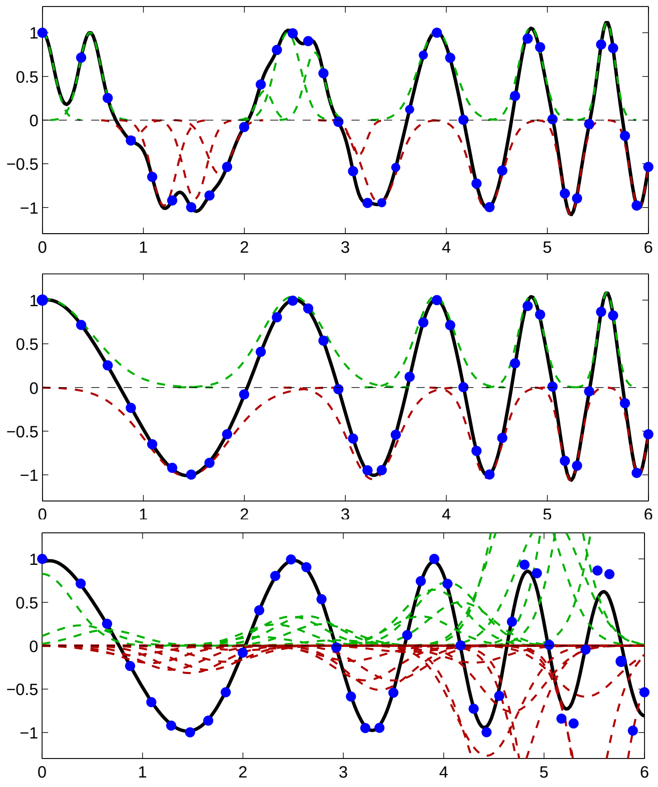

Fig. 3. Gaussian mixture fitting on a general (positive/negative) function: case $\lambda = 0$, $\delta > 0$ (top), case λ, $\delta > 0$ (middle), and Gaussian SVR (bottom). In each picture: training samples (dots), individual Gaussian elements (dashed), and resulting Gaussian mixture (solid)

The results after 50 iterations are shown in Figure 3 from top to bottom respectively. The first case, with $\lambda = 0$, shows clear signs of over fitting as it reaches a situation of negligible fitting error. On the contrary, the regularization on the precision promotes "wider" Gaussian functions that compete against each other for the data; the solution results in an optimal number of relevant Gaussian elements, equal to the number of lobes in the chirp signal. Finally, the performance of the SVR is well known to depend on the selection of the Gaussian kernel variance; the variance selected here yields poor fitting on the narrower chirp lobes; obviously, one could have selected a smaller variance to improve the fitting on that region; in any case, SVR's factor of utilization[9] in this scenario never drops below 100%.

The second scenario corresponds to the identification of the two-dimensional mexican *sombrero* function, defined by

$$y(x_1, x_2) = \text{sinc}\left(3\sqrt{x_1^2 + x_2^2}\right) \tag{51}$$

where $\text{sinc}(a) = \sin(a)/(\pi a)$. Given that this function has circular symmetry, its identification with a sparse Gaussian mixture is somewhat challenging. The dataset was obtained by selecting a random point inside every cell of a regular 9-segment cartesian division within the interval $[-3, 3]$; no noise was added to the data. A Gaussian mixture of $K = 81$ elements, equal to the training dataset size ($K = L$), was used in the experiment. The previous three methods were used here; in all cases, the precision was initially set to a value of 3 (in SVR, this value is a hyper-parameter, and thus fixed for all Gaussian kernels). The results after 50 iterations are shown in Figure 4. The outcome of this experiment follows:

1) the weight regularization ($\lambda = 0$, $\delta > 0$) cannot defeat over-fitting, and the utilization falls to 52%,
2) the method with regularization in the precision ($\lambda > 0$, $\delta > 0$) achieves good learning and generalization performance with 34% of the Gaussian elements,
3) the SVR presents similar performance than the previous method, but with a 100% utilization.

In the previous qualitative experiments, learning and generalization has been assessed visually. In the next synthetic scenario, a formal methodology, 10-fold cross validation, is used: the dataset is randomly divided into 10 blocks of data; for every block, each method is trained on the remaining blocks and tested on the hold-out block; results, averaged over all test blocks, thus reflect predictive performance of unseen cases. The experiment deals with the identification of

[9]The factor of utilization refers to the percentage of Gaussian elements that result in non-zero weight after training.

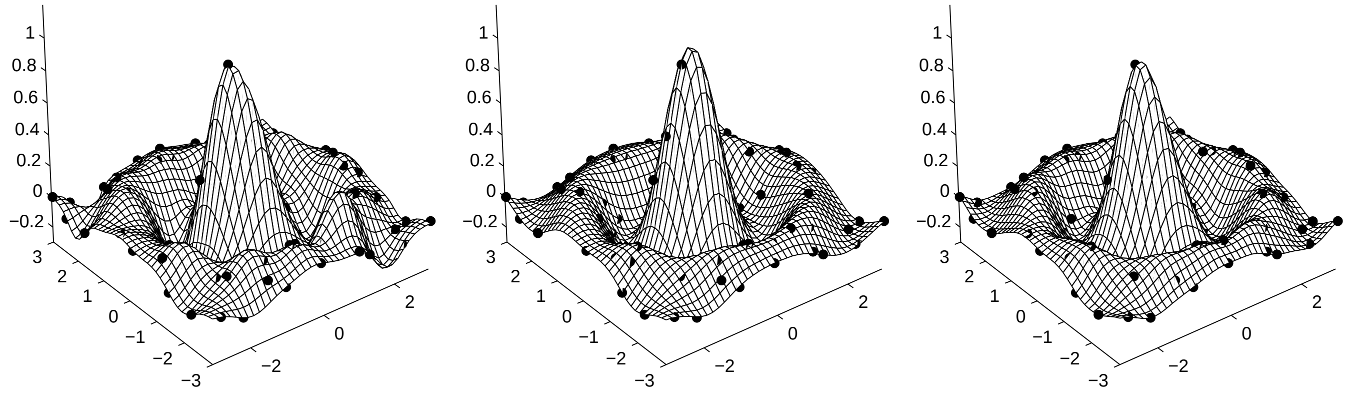

Fig. 4. Identification of the mexican *sombrero* function (51). Propose method for $\lambda = 0$, $\delta > 0$ (left), λ, $\delta > 0$ (middle), and Gaussian SVR (right).

the deterministic component of the four-dimensional variant of the `add10` function suggested in the DELVE database

$$y(\mathbf{x}) = 5 + 5\sin(3\pi x_1 x_2) + 10(x_3 - 0.5)^2 + 5x_4 + \eta \quad (52)$$

where $x_i \in [0,\ 1]$ and η is zero-mean Gaussian noise. The dataset corresponds to random points inside every hypercell of a regular 7-segment cartesian division; this procedure results in $L = 7^4 = 2401$ points, at which the function (52) was evaluated. Given that the function is positive in the range of interest, we selected the original algorithm for non-negative functions detailed in Table 1: the number of Gaussian elements was $K = 40$; as the space dimension is $d = 4$, the number of parameters in each Gaussian becomes $N = 15$. The LS-SVR and ϵ-insensitive SVR [13] were selected as comparative methods. The precision of the Gaussian kernel was obtained by exhaustively assessing with cross validation the average LS-SVR performance on the clean and noisy datasets; the best performance resulted with a value of precision equal to 7.

The interpolation and the regression ability of each method was tested on the purely deterministic data and a noisy version thereof (η with variance equal to 1) respectively. The result of this experiment after 100 iterations is brought in Table I. The values correspond to the normalized MSE I, i.e., the ratio between the MSE of the method and the variance of the reference $y(\mathbf{x})$

$$I = \frac{\sum_{\mathbf{x}} \left(y(\mathbf{x}) - f(\mathbf{x})\right)^2}{\sum_{\mathbf{x}} \left(y(\mathbf{x}) - \overline{y}\right)^2}. \quad (53)$$

where $\overline{y} = \sum_{\mathbf{x}} y(\mathbf{x}) / \sum_{\mathbf{x}} 1$ is the average value of the reference function. The column "Lin" corresponds to the least-squares linear fit. The 10-fold cross validation delivers two values, namely, the *testing* error and the *training* error: these two values close to each other indicates good generalization abilities.

	GFM		LS-SVR		ϵ-SVR	Lin
η^2	I_{test}	I_{train}	I_{test}	I_{train}	I_{test}	I_{test}
0	$8.4e^{-3}$	$6.6e^{-3}$	$\mathbf{1.6e^{-3}}$	$3.7e^{-4}$	$1.7e^{-2}$	0.76
1	$\mathbf{9.6e^{-2}}$	$7.2e^{-2}$	$1.1e^{-1}$	$3.9e^{-2}$	0.42	0.79

TABLE I
NORMALIZED MEAN SQUARE ERROR IN THE `ADD10` DATASET.

The proposed GFM method presents excellent generalization and learning capabilities. Given the low number of Gaussian functions ($K = 40$), the final utilization results in 100%. LS-SVR performance in this scenario is also excellent,[10]: in the noiseless case its learning capabilities are somewhat ahead of the proposed GFM, but in the regression test, SVR shows signs of overfitting, falling slightly behind GFM. However, the major difference arises when comparing the number of parameters to estimate: the proposed GFM deals with only $KN = 600$ parameters, while SVR complexity is equal to the size of the dataset $L = 2401$.

[10]On the contrary, ϵ-SVR yields poor performance because the stochastic component in the data is not heavy-tailed; its numerical optimization method turned out also computationally prohibitive.

B. Real Scenarios

We consider now selected regression datasets from the DELVE and UCI repositories, in particular, `kin8`, `pumadyn8`, `abalone` and `energy` (heating and cooling) [32]. In all cases, the dimension of the input space is equal to $d = 8$. The reference $y(\mathbf{x})$ in the `kin8`, `abalone` and `energy` datasets was non-negative. Therefore, two variants of the proposed regression methodology were tested:

1) the identification of the positive function $y(\mathbf{x})$ with the original method (GM) for non-negative functions, detailed in Sec. II,
2) the identification of the function $y(\mathbf{x}) - b$ with the general method (GM$\pm$) detailed in Sec. IV, where b is equal to the average value of the reference.

On the other hand, support vector regression with quadratic loss was used as main comparative method. The precision of the Gaussian kernel was obtained by exhaustively assessing with cross validation the average LS-SVR performance on every dataset (in case of `kin` and `pumadyn`, the value was selected from the average performance for the high "`h`" and medium "`m`" noise).[11] In every scenario, the number of Gaussian functions is $K = 50$: as the number of parameters of each Gaussian is $N = 45$, the total number of system parameters results thus in 2,250.

Table II contains the result of these experiments with 10-fold cross validation averaged over ten runs. The performance of two additional state-of-the-art methods (codenamed M1 and M2) is also included: CMAC-based regression [30] and a regression tree [31] were selected for both `kin` and `pumadyn`, and linear regression robust to outliers (least absolute error) and a regression random forest [32], [33] for `abalone` and `energy`.

For each column the best performance has been highlighted in boldface. Except for the case `kin8f` "fair linearity", the proposed GFM method outperforms SVR in all other scenarios. In some cases the differences are not very significative, but the results confirm the proposed approach as an interesting alternative to be considered. In the scenario `energy`, proposed recently [32], GFM outperforms the random forest by nearly an order of magnitude. It is also interesting to note that there the size of the dataset (768) is much lower than the GFM size N, yet overfitting does not occur. Finally, selecting GFM or GFM$\pm$ in an specific scenario is nearly arbitrary as the performance seems to be comparable. Our suggestion though is to use GFM when the data has an obvious non-negative nature (such as in `abalone` and `energy`).

VI. DISCUSSION

The computational complexity of the proposed method is dominated by the solution of the K linear equation systems in (23). The cost of each one of these problems is of the order $\mathcal{O}(d^6)$ and therefore the overall computational cost per iteration is $\mathcal{O}(Kd^6)$. The number of iterations until convergence can depend on several factors such as the GLUF

[11]Furthermore, given the rigidity of SVR kernel, the raw input data and the input data normalised in the interval $[0,\ 1]$ have been tested, having selected whichever leads to better performance.

	kin8nm	kin8nh	kin8fm	kin8fh	puma8nm	puma8nh	abalone	heat	cool
Optimal	$4.1\mathrm{e}^{-2}$	$3.0\mathrm{e}^{-1}$	$2.2\mathrm{e}^{-2}$	$2.5\mathrm{e}^{-1}$	$2.9e^{-2}$	$2.5e^{-1}$	—	—	—
Lin	$6.7\mathrm{e}^{-1}$	$8.1\mathrm{e}^{-1}$	$6.9\mathrm{e}^{-2}$	$3.1\mathrm{e}^{-1}$	$4.8e^{-1}$	$6.3\mathrm{e}^{-1}$	$5.3e^{-1}$	$9.2e^{-2}$	$2.0e^{-1}$
GFM	$1.6\mathrm{e}^{-1}$	$5.0\mathrm{e}^{-1}$	$5.4\mathrm{e}^{-2}$	$2.9\mathrm{e}^{-1}$	$\boldsymbol{3.9e^{-2}}$	$\mathbf{3.3e^{-1}}$	$\boldsymbol{4.3e^{-1}}$	$\boldsymbol{3.6e^{-3}}$	$\boldsymbol{2.1e^{-2}}$
GFM±	$\mathbf{8.9e^{-2}}$	$\mathbf{3.3e^{-1}}$	$4.7\mathrm{e}^{-2}$	$3.1\mathrm{e}^{-1}$	$4.5e^{-2}$	$3.4e^{-1}$	$4.3e^{-1}$	$7.8e^{-3}$	$3.1e^{-2}$
LS-SVR	$1.8\mathrm{e}^{-1}$	$6.3\mathrm{e}^{-1}$	$\mathbf{2.7e^{-2}}$	$\mathbf{2.6e^{-1}}$	$4.6e^{-2}$	$5.5e^{-1}$	$\boldsymbol{4.3e^{-1}}$	$5.2\mathrm{e}^{-3}$	$2.3e^{-2}$
M1	$9.1\mathrm{e}^{-2}$	$3.3\mathrm{e}^{-1}$	$9.5\mathrm{e}^{-2}$	$3.0\mathrm{e}^{-1}$	$4.0e^{-2}$	$3.3e^{-1}$	$5.1e^{-1}$	$9.7\mathrm{e}^{-2}$	$1.3e^{-1}$
M2	$4.5\mathrm{e}^{-1}$	$6.1\mathrm{e}^{-1}$	$2.0\mathrm{e}^{-1}$	$4.3\mathrm{e}^{-1}$	$5.0e^{-2}$	$3.4e^{-2}$	$4.4e^{-1}$	$1.0\mathrm{e}^{-2}$	$7.3e^{-2}$

TABLE II
PERFORMANCE ANALYSIS ON DATASETS FROM DELVE AND UCI REPOSITORIES.

bias σ, the dimensionality of the problem, and the hyper-parameter μ to ensure the proper conditioning of the successive optimization problems. Although a thorough analysis of the convergence properties is beyond the scope of this paper, we must mention that the overall speed of the proposed technique can be reduced, for instance, by using regularization techniques beyond the diagonal loading (as suggested in Sec. II-C), by parallelizing the solution of the K linear systems, by relaxing the precision of the intermediate solutions of the linear equations systems, or by taking into account that the solutions of (23) in consecutive iterations should not be very different, that is, we can use a warm-start approach for speeding up the solution of the linear equation systems.

The results presented on this paper refer to the method based on equation (21), which corresponds to a modification of the original equation (20) with the aim of including the (positive-definite term of the) Hessian as system matrix, therewith improving the speed of convergence. Fig. 5 validates this choice on the experiments of Table I: on the noiseless scenario ($\eta = 0$), the proposed alternative (21) delivers steady learning performance, while (20) stalls at some point during the training. Apart from the case of perfect fitting, in which both equations can share the same solution, an analytical study on the differences/connections between both does not seem easy. Based on empirical evidence, we thus firmly suggest to include the *square* of the relevance as in (21).

Finally, given that the parametric complexity of the general multivariate Gaussian functions grows (5) with the square of the space dimension d, its use in problems with very large dimensions turns out problematic. Therefore, another way to alleviate the computational complexity in the regression machine is namely to reduce its basic parametric size N. Some options worth taking into consideration are:

1) a diagonal precision matrix (with only d parameters),
2) a Gaussian radial basis function, with the same precision in all dimensions (and thus a single precision value),
3) the center set to a fixed value, such as a data point.

Obviously, the reduction in degrees of freedom implies that the identification capabilities of each function shrink. This option is likely to have a twofold effect though, namely, in theory the capacity of the machine reduces accordingly and its generalization performance must improve therewith. The dilemma that immediately arises is whether a full-fledged GFM with few elements is a better option than a larger mixture of simpler Gaussian functions. In that sense, the combination of the last two options above, namely a radial basis function (single-valued parametric precision) with its center fixed to a data point, is an interesting option as it compares seamlessly to support vector machines, having in addition the flexibility of a parametric Gaussian kernel variance/precision. Deep study on these alternatives is a promising exciting research avenue worth undertaking in the future.

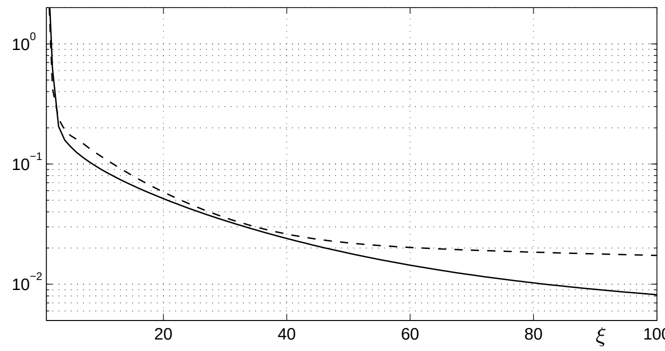

Fig. 5. Speed of convergence for the iterative method based on the solution equation (20) (dashed) and that of equation (21) (solid).

VII. CONCLUSIONS

Multivariate Gaussian mixture approximation driven by the mean square error (MSE) criterion is well-known to result in a highly non-linear and non-convex problem. The generalized logarithmic utility function (GLUF) has allowed us to revamp this challenging problem into another one whose gradient and Hessian resembles that of a "locally" convex problem. The numerical implementation results in several iterative least-squares inversion problems, each one involving the parameters of each individual multivariate Gaussian function. In consequence, the proposed method can be optimized for running in parallel-computing hardware. The method, devised initially for non-negative reference functions, has been easily extended to general functions. Its performance has been explored on synthetic and real scenarios, delivering very competitive generalization and learning abilities.

REFERENCES

[1] J. Park and I. Sandberg, "Universal approximation using radial-basis function networks," *Neural Comput.*, vol. 3, pp. 246–257, Jun. 1991.
[2] L. Cohen, *Time–frequency analysis*. Prentice Hall, 1995.
[3] P. Yee and S. Haykin, "A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2503–2521, Sep. 1999.
[4] S. Seshagiri and H. Khalil, "Output feedback control of nonlinear systems using RBF neural networks," *IEEE Trans. Neural Netw.*, vol. 11, pp. 69–79, Jan. 2000.

[5] T. Blu and M. Unser, “Wavelets, fractals, and radial basis functions,” *IEEE Trans. Signal Process.*, vol. 50, pp. 543–553, Mar. 2002.

[6] M. Yee, B. Yeap, and L. Hanzo, “Radial basis function-assisted turbo equalization” *IEEE Trans. Commun.*, vol. 51, pp. 664–675, Apr. 2003.

[7] N. Gkalelis, V. Mezaris, I. Kompatsiaris, and T. Stathaki, “Mixture subclass discriminant analysis link to restricted Gaussian model and other generalizations,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 1, pp. 8–21, Jan. 2013.

[8] C. Tseng and S. Lee, “Design of fractional order digital differentiator using radial basis function,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 7, pp. 1708–1718, Jul. 2010.

[9] J. Jacobs, “Bayesian support vector regression with automatic relevance determination kernel for modelling of antenna input characteristics,” *IEEE Trans. Antennas Propag.*, vol. 60, no. 4, pp. 2114–2118, Apr. 2012.

[10] L. Weruaga, ”Redundant time–frequency marginals for chirplet decomposition,” *IEEE Intl. Wksp. Mach. Learn. Signal Process.* 2012, pp. 1–5.

[11] S. Chen, C. Cowan, and P. Grant, “Orthogonal least squares learning algorithm for radial basis function networks,” *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[12] P. Yee and S. Haykin. *Regularized radial basis function networks: Theory and applications*, Wiley, 2001.

[13] B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

[14] S. O. Haykin, *Neural networks and learning machines (3rd Ed.),* Prentice Hall, Upper Saddle River, NJ, 2008.

[15] N. Karayiannis, “Reformulated radial basis neural networks trained by gradient descent,” *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 657–671, May 1999.

[16] D. Simon, “Training radial basis neural networks with the extended Kalman filter,” *Neurocomp.*, vol. 48, pp. 455–475, 2002.

[17] G. Huang, P. Saratchandran, and N. Sundararajan, “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation,” *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 57–67, Jan. 2005.

[18] M. Bortman and M. Aladjem, “A growing and pruning method for radial basis function networks,” *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1039–1045, Jun. 2009.

[19] J. Lian, Y. Lee, S. Sudhoff, and S. Zak, “Self-organizing radial basis function network for real-time approximation of continuous-time dynamical systems,” *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 460–474, Mar. 2008.

[20] S. Chen, X. Hong, B. Luk, and C. Harris, “Construction of tuneable radial basis function networks using orthogonal forward selection,” *IEEE Syst., Man, Cybern.*, vol. 39, no. 2, pp. 457–466, Apr. 2009.

[21] H. Huan, D. Hien, and H. Tue, “Efficient algorithm for training interpolation RBF networks with equally spaced nodes,” *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 982–988, Jun. 2011.

[22] T. Xie, H. Yu, J. Hewlett, P. Rózycki, and B. Wilamowski, “Fast and efficient second-order method for training radial basis function networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 609–619, Apr. 2012.

[23] C. Friedman, *Utility-based learning from data,* CRC Press, 2010.

[24] H. Guo, “A simple algorithm for fitting a Gaussian function,” *IEEE Signal Process. Mag.*, vol. 28, no. 5, pp. 134–137, Sep. 2011.

[25] S. Boyd and L. Vandenberghe, *Convex optimization,* Cambridge University Press, 2004.

[26] J. Gu and P. J. Wolfe “Robust adaptive beamforming using variable loading,” *IEEE Wksp. Sensor Array Multichan. Process.* 2006, pp. 1–5.

[27] J. A. Bilmes, “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models,” Tech. Rep. 97-021, *Intl. Comp. Sci. Inst.*, Berkeley CA, 1997.

[28] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006,

[29] K. Hlaváčková-Schindler, “Tikhonov regularization parameter in reproducing kernel Hilbert spaces with respect to the sensitivity of the solution,” *Intl. Conf. Artif. Neural Netw. (ICANN)* 2008, pp. 215–224.

[30] L. Weruaga and B. Kieslinger, “Tikhonov training of the CMAC neural network,” *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 613–622, May 2006.

[31] Y. Morimoto, H. Ishii, and S. Morishita, “Efficient construction of regression trees with range and region splitting,” *Mach. Learn.*, vol. 45, no. 3, pp. 235–259, Dec. 2001.

[32] A. Tsanasa and A. Xifara, “Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools,” *Energy Buildings*, vol. 49, pp. 560–567, Jun. 2012.

[33] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

PLACE PHOTO HERE

Luis Weruaga (M’95, SM’11) received the M.S. degree in telecommunications engineering in 1990 from the University of Vigo, and the Ph.D. in the same field in 1994 from the Polytechnic University of Madrid, Spain. From 1995 to 1998 he worked for Telefonica Spain as a R&D engineer, period in which he co-worked in Graphnet Inc. (Teaneck, NJ) for one year. In 1998 he joined Sema Group Spain as head of its DSP unit, becoming Assist. Prof. in *signal and communications theory* at Cartagena University of Technology, Spain one year later. In 2003 he joined the Austrian Academy of Sciences, Vienna, where he worked for almost seven years. During that time, he was also adjunct lecturer at the Technical University of Vienna. Since 2010 he is Assoc. Prof. at Khalifa University (KUSTAR), United Arab Emirates. He has published more than sixty papers in conference and peer-reviewed journals, and holds two patents. His main technical interest are time–frequency analysis and machine learning.

PLACE PHOTO HERE

Javier Vía (M’08, SM’12) received his telecommunication engineer degree and his Ph.D. in electrical engineering from the University of Cantabria, Spain in 2002 and 2007, respectively. In 2002 he joined the Department of Communications Engineering, University of Cantabria, Spain, where he is currently Associate Professor. He has spent visiting periods at the Smart Antennas Research Group of Stanford University, and at the Department of Electronics and Computer Engineering (Hong Kong University of Science and Technology). Dr. Vía has actively participated in several European and Spanish research projects. His current research interests include blind channel estimation and equalization in wireless communication systems, multivariate statistical analysis, quaternion signal processing and kernel methods.