Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 627372, 12 pages
doi:10.1155/2010/627372

Research Article

A Decentralized Approach for Nonlinear Prediction of Time Series Data in Sensor Networks

Paul Honeine (EURASIP Member),[1] Cédric Richard,[2] José Carlos M. Bermudez,[3] Jie Chen,[2] and Hichem Snoussi[1]

[1] *Institut Charles Delaunay, Université de Technologie de Troyes, 6279 UMR CNRS, 12 rue Marie Curie, BP2060, 10010 Troyes Cedex, France*
[2] *Fizeau Laboratory, Observatoire de la Côte d'Azur, Université de Nice Sophia-Antipolis, 6525 UMR CNRS, 06108 Nice, France*
[3] *Department of Electrical Engineering, Federal University of Santa Catarina, 88040-900 Florianópolis, SC, Brazil*

Correspondence should be addressed to Paul Honeine, paul.honeine@utt.fr

Received 30 October 2009; Revised 8 April 2010; Accepted 9 May 2010

Academic Editor: Xinbing Wang

Wireless sensor networks rely on sensor devices deployed in an environment to support sensing and monitoring, including temperature, humidity, motion, and acoustic. Here, we propose a new approach to model physical phenomena and track their evolution by taking advantage of the recent developments of pattern recognition for nonlinear functional learning. These methods are, however, not suitable for distributed learning in sensor networks as the order of models scales linearly with the number of deployed sensors and measurements. In order to circumvent this drawback, we propose to design reduced order models by using an easy to compute sparsification criterion. We also propose a kernel-based least-mean-square algorithm for updating the model parameters using data collected by each sensor. The relevance of our approach is illustrated by two applications that consist of estimating a temperature distribution and tracking its evolution over time.

1. Introduction

Wireless sensor networks consist of spatially distributed autonomous sensors whose objective is to cooperatively monitor physical or environmental parameters such as temperature, humidity, concentration, pressure, and so forth. Starting with critical military applications, they are now used in many industrial and civilian areas, including industrial process monitoring and control, environment and habitat monitoring, home automation, and so forth. Some common examples are monitoring the state of permafrost and glacier, tracking wildland and forest fires spreading, detecting water and air pollution, sensing seismic activities, to mention a few. Modeling phenomena under consideration allows the extrapolation of present states over time and space. This can be used to identify trends or to estimate uncertainties in forecasts, and to prescribe detection, prevention, or control strategies accordingly.

Here, we consider the problem of modeling complex processes such as heat conduction and pollutant diffusion with wireless sensor networks, and track changes over time and space. This typically leads to a dilemma between incorporating enough complexity or realism on the one hand, and keeping the model tractable on the other hand. Due to computational resource limitations, priority is given to oversimplification and models that can separate the important from the irrelevant. Many approaches have been proposed to address this issue with collaborative sensor networks. In [1], an incremental subgradient optimization procedure has been applied in a distributed fashion for the estimation of a single parameter. See also [2] for an extension to clusters. It consists of passing the parameter from sensor to sensor, and updating it to minimize a given cost function locally. More than one pass over all the sensors may be required for convergence to the optimal (centralized) solution. This number of cycles can be theoretically bounded. The main advantages of this method are the simple sensor-to-sensor scheme with a short pathway and without lateral communication, and the need to communicate only the estimated parameter value over sensors. However, as explained in [3], such a

technique cannot be used for functional estimation since evaluating the subgradient in the vicinity of each sensor requires information related to other sensors. Model-based techniques that exploit the temporal and spatial redundancy of data in order to compress communications have also been considered. For instance, in [4], data captured by each sensor over a time interval are fitted by (cubic) polynomial curves whose coefficients are communicated between sensors. Since there is a significant amount of redundancy between measurements performed by two nearby sensors, spatial correlations are also modeled by defining the basis functions over both spatial parameters and time. The main drawback of such techniques is their dependence upon the modeling assumptions. Model-independent methods based on kernel machines have recently been investigated. In particular, a distributed learning strategy has been successfully applied to regression in sensor networks [5, 6]. Here, each sensor acquires information from neighboring sensors to solve locally the least-squares problem. This broadcast, unfortunately, leads to high energy consumption.

We take advantage of the pros of some of the abovementioned methods to derive our approach. In particular, we will require the following important properties to hold.

(i) Sensor-to-sensor scheme: each sensor has the same importance in the network at each updating cycle. Thus, failure of any sensor has a small impact on the overall model, as opposed to cluster-head failure in aggregation and clustering techniques. It should be noted that several such conventional methods have been investigated specifically for use in sensor networks. Examples are LEACH with data fusion in the cluster head [7], PEGASIS with data conveyed to a leader sensor [8], and (minimum) spanning tree and junction tree, to name a few.

(ii) Kernel machines: these model-independent methods have gained popularity over the last decade. Initially derived for regression and classification with *support vector machines* [9], they include classical techniques such as least-squares methods and extend them to nonlinear functional approximation. Kernel machines are increasingly applied in the field of sensor networks for localization [10], detection [11], and regression [3]. One potential problem of applying classical kernel machines to distributed learning in sensor networks is that the order of the resulting models scales linearly with the number of deployed sensors and measurements.

(iii) Spatial redundancy: taking spatial correlation of data into account has been recommended by numerous researchers. See, for example, [12–14] where relation between the topology of the network and measurement data is studied. In particular, the authors of [12] seek to identify a small subset of representative sensors which leads to minimal distortion of the data.

In this paper, we propose a new approach to model physical phenomena and track their evolution over time. The new approach is based on a kernel machine but controls the model order through a coherence-based criterion that reduces spatial redundancy. It also employs sensor-to-sensor communication, and thus is robust to single sensor failures. The paper is organized as follows. The next section briefly reviews functional learning with kernel machines and addresses its limitations within the context of wireless sensor networks. It is shown how to overcome these limitations through a model order reduction strategy. Section 3 describes the proposed algorithm and its application to instantaneous functional estimation and tracking. Section 4 addresses implementation issues in sensor networks. Finally, we report simulation results in Section 5 to illustrate the applicability of the proposed approach.

2. Functional Learning and Sensor Networks

We consider a regression problem whose goal is, for example, to estimate the temperature distribution over a region where n wireless sensors are randomly deployed. We denote by $\mathcal{X}$ the region of interest, which is supposed to be a compact subset of $\mathbb{R}^d$, and by $\|\cdot\|$ its conventional Euclidean norm. We wish to determine a function $\psi^*(\cdot)$ defined on $\mathcal{X}$ that best models the spatial temperature distribution. The latter is learned from the information coupling sensor locations and measurements. The information from the n sensors located at $\mathbf{x}_i \in \mathcal{X}$ and providing measurements $d_i \in \mathbb{R}$, with $i = 1,\ldots,n$, is combined in the vector of pairs $\{(\mathbf{x}_1, d_1),\ldots,(\mathbf{x}_n, d_n)\}$. The fitness criterion is the mean square error between the model outputs $\psi(\mathbf{x}_i)$ and the measurements d_i, for $i = 1,\ldots,n$, namely,

$$\psi^*(\cdot) = \arg\min_{\psi} \frac{1}{n}\sum_{i=1}^{n}\left(d_i - \psi(\mathbf{x}_i)\right)^2. \tag{1}$$

Note that this problem is underdetermined since there exists an infinite number of functions that verify this expression. To obtain a well-posed problem, one must restrict the space of candidate functions. The framework of reproducing kernels allows us to circumvent this drawback.

2.1. A Brief Review of Kernel Machines. We consider a reproducing kernel $\kappa : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. We denote by $\mathcal{H}$ its reproducing kernel Hilbert space, and by $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ the inner product in $\mathcal{H}$. This means that every function $\psi(\cdot)$ of $\mathcal{H}$ can be evaluated at any $\mathbf{x} \in \mathcal{X}$ with

$$\psi(\mathbf{x}) = \left\langle \psi(\cdot), \kappa(\cdot,\mathbf{x})\right\rangle_{\mathcal{H}}. \tag{2}$$

By using Tikhonov regularization, minimization of the cost functional over $\mathcal{H}$ leads to the optimization problem

$$\psi^*(\cdot) = \arg\min_{\psi\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}\left(d_i - \psi(\mathbf{x}_i)\right)^2 + \eta\ ||\psi||^2_{\mathcal{H}}, \tag{3}$$

where η controls the trade-off between the fitting to the available data and the smoothness of the solution.

Before proceeding, we recall that data-driven reproducing kernels have been proposed in the literature, as well as

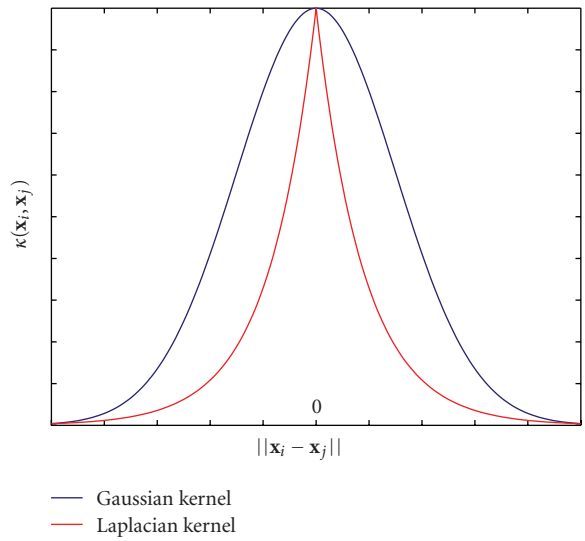


FIGURE 1: Shapes of the Gaussian and the Laplacian kernels around the origin.

more classical and universal ones. In this paper, without any essential loss of generality, we are primarily interested in radial kernels. They can be expressed as a decreasing function of the Euclidean distance in $\mathcal{X}$, that is, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\|\mathbf{x}_i - \mathbf{x}_j\|)$ with some abuse of notation. Radial kernels have a natural interpretation in terms of measure of similarity in $\mathcal{X}$, as the kernel value is larger the closer together two locations are. Two typical examples of radial kernels are the Gaussian and Laplacian kernels, defined as

$$\begin{aligned} \text{Gaussiankernel:} \quad & \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\beta_0^2} \\ \text{Laplaciankernel:} \quad & \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\| / \beta_0}, \end{aligned} \tag{4}$$

where β_0 is the kernel bandwidth. Figure 1 represents these kernels. Other examples of kernels, radial or not, can be found in [15].

It is well-known in the machine-learning community that the optimal solution of the optimization problem (3) can be written as a kernel expansion in terms of the available data [16, 17], namely,

$$\psi^*(\cdot) = \sum_{k=1}^{n} \alpha_k \, \kappa(\mathbf{x}_k, \cdot). \tag{5}$$

This means that the optimal function is uniquely identified by the weighting coefficients $\alpha_1, \ldots, \alpha_n$ and the n sensor locations $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Whereas the initial optimization problem (3) considers the infinite dimensional hypothesis space $\mathcal{H}$, we are now considering the optimal vector $\boldsymbol{\alpha} = [\alpha_1 \cdots \alpha_n]^\top$ in the n-dimensional space of coefficients. The corresponding cost function is obtained by inserting the model (5) into the optimization problem (3). This yields

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{d} - \mathbf{K}\boldsymbol{\alpha}\|^2 + \eta \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}, \tag{6}$$

where $\mathbf{K}$ is the so-called Gram matrix whose (i, j)th entry is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{d} = [d_1 \ \ldots \ d_n]^\top$ is the vector of measurements. The solution to this problem is given by

$$\boldsymbol{\alpha}^* = \left(\mathbf{K}^\top \mathbf{K} + \eta \mathbf{K}\right)^{-1} \mathbf{K}^\top \mathbf{d}. \tag{7}$$

Note that the computational complexity involved in solving this problem is $\mathcal{O}(n^3)$.

Practicality of wireless sensor networks imposes constraints on the computational complexity of calculations performed by each sensor, and on the amount of internode communications. To deal with these constraints, the optimization problem (6) may be solved distributively using a receive—update—transmit scheme. For example, sensor i gets the parameter vector $\boldsymbol{\alpha}_{i-1}$ from sensor $i-1$, and updates it to $\boldsymbol{\alpha}_i$ based on the error e_i defined by

$$\begin{aligned} e_i &= d_i - [\kappa(\mathbf{x}_1, \mathbf{x}_i) \cdots \kappa(\mathbf{x}_n, \mathbf{x}_i)] \boldsymbol{\alpha}_i \\ &= d_i - \psi(\mathbf{x}_i). \end{aligned} \tag{8}$$

In order to compute $[\kappa(\mathbf{x}_1, \mathbf{x}_i) \cdots \kappa(\mathbf{x}_n, \mathbf{x}_i)]$; however, each sensor must know the locations of the other sensors. This unfortunately imposes a substantial demand for both storage and computational time, as most practical applications require a large number of densely deployed sensors for coverage and robustness reasons. To alleviate these constraints, we propose to control the model complexity to significantly reduce the computational effort and communication requirements.

2.2. Complexity Control of Kernel Machines in Sensor Networks. Consider the restriction of the kernel expansion (5) to a dictionary $\mathcal{D}_m$ composed of m functions $\kappa(\mathbf{x}_{\omega_k}, \cdot)$ carefully selected among the n available ones, where $\{\omega_1, \ldots, \omega_m\}$ is a subset of $\{1, \ldots, n\}$ and m is several orders of magnitude smaller than n. This is equivalent to choosing m sensors denoted by $\omega_1, \ldots, \omega_m$ whose locations are given by $\mathbf{x}_{\omega_1}, \ldots, \mathbf{x}_{\omega_m}$. The resulting reduced order model will be given by

$$\psi(\cdot) = \sum_{k=1}^{m} \alpha_k \kappa(\mathbf{x}_{\omega_k}, \cdot). \tag{9}$$

The selection of the kernel functions in the reduced-order model is crucial for achieving good performance. In particular, the removed kernel functions must be well approximated by the remaining ones in order to minimize the difference between the optimal model given in (5) and the reduced one in (9). A variety of methods have been proposed in recent years for deriving kernel-based models with reduced order. They broadly fall into two categories. In the first one, the optimization problem (6) is regularized by an ℓ_1 penalization term applied to $\boldsymbol{\alpha}$ [18, 19]. These techniques are not suitable for sensor networks due to their large computational requirements. In the second category, postprocessing algorithms are used to control the model order when new data becomes available. For instance, the short-time approach consists of including,

as we visit each sensor, the newly available kernel function while removing the oldest one. Another technique, called truncation, removes the kernel functions associated with the smallest weighting coefficients α_i. These naive methods usually exhibit poor performance because they ignore the relationships between the kernel functions of the model. To efficiently control the order of the model (9) as the model travels through the network, only the less redundant kernel functions must be added to the kernel expansion. Several criteria have been proposed in the literature to assess the contribution of each new kernel function to an existing model. In [20–22], for instance, the kernel function $\kappa(\mathbf{x}_i, \cdot)$ is inserted into the model and its order is increased by one if the approximation error defined below is greater than a given threshold ϵ

$$\min_{\beta_1,\ldots,\beta_m} \left\| \kappa(\mathbf{x}_i, \cdot) - \sum_{k=1}^{m} \beta_k \kappa(\mathbf{x}_{\omega_k}, \cdot) \right\|_{\mathcal{H}}^2 \geq \epsilon, \tag{10}$$

where κ is a unit-norm kernel (replace $\kappa(\mathbf{x}_i, \cdot)$ with $\kappa(\mathbf{x}_i, \cdot)/\sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i)}$ in (10) if $\kappa(\mathbf{x}_i, \cdot)$ is not unit-norm.) that is, $\kappa(\mathbf{x}_i, \mathbf{x}_i) = 1$. Note that this criterion requires the inversion of an m-by-m matrix, and thus demands high precision and large computational effort from the microprocessor at each sensor.

In this paper, we cut down the computational cost associated with this selection criterion by using an approximation which has a natural interpretation in the wireless sensor network setting. Based on recent work in kernel-based online prediction of time series by three of the authors [23, 24], we employ the coherence criterion which includes the candidate kernel function $\kappa(\mathbf{x}_i, \cdot)$ in the mth order model provided that

$$\max_{k=1,\ldots,m} |\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k})| \leq \nu, \tag{11}$$

where ν is a threshold in $[0, 1[$ which determines the sparsity level of the model. By the reproducing property of $\mathcal{H}$, we note that $\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k}) = \langle \kappa(\mathbf{x}_i, \cdot), \kappa(\cdot, \mathbf{x}_{\omega_k}) \rangle_{\mathcal{H}}$. The condition (11) then results in a bounded crosscorrelation of the kernel functions in the model. Without going into details, we refer interested readers to our recent paper [23], where we study the properties of the resulting models, and connections to other sparsification criteria such as (10) or the kernel principal component analysis.

We shall now show that the coherence criterion has a natural interpretation in the wireless sensor network setting. Let us compute the distance of two kernel functions in $\mathcal{H}$

$$\begin{aligned} &\left\| \kappa(\mathbf{x}_i, \cdot) - \kappa\left(\mathbf{x}_j, \cdot\right) \right\|_{\mathcal{H}}^2 \\ &\quad = \left\langle \kappa(\mathbf{x}_i, \cdot) - \kappa\left(\mathbf{x}_j, \cdot\right), \kappa(\mathbf{x}_i, \cdot) - \kappa\left(\mathbf{x}_j, \cdot\right) \right\rangle_{\mathcal{H}} \\ &\quad = 2\left(1 - \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)\right), \end{aligned} \tag{12}$$

where we have assumed, without substantive loss of generality, that κ is a unit-norm kernel. Back to the coherence criterion and using the above result, (11) can be written as follows:

$$\min_{k=1,\ldots,m} \|\kappa(\mathbf{x}_i, \cdot) - \kappa(\mathbf{x}_{\omega_k}, \cdot)\|_{\mathcal{H}}^2 \geq 2(1 - \nu). \tag{13}$$

Table 1: Distributed learning algorithm.

In-sensor parameters	
Evaluation of the kernel	$\kappa(\cdot\,, \cdot)$
Coherence threshold	ν
Step-size parameter	ρ
Communicated message	
Locations of selected sensors	$[\mathbf{x}_{\omega_1} \cdots \mathbf{x}_{\omega_m}]$
Weighting coefficients	$[\alpha_{\omega_1,i-1} \cdots \alpha_{\omega_m,i-1}] = \boldsymbol{\alpha}_{i-1}^{\top}$
At each sensor i	
(1) Compute $\boldsymbol{\kappa}_i$	$\boldsymbol{\kappa}_i = [\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_1}) \cdots \kappa(\mathbf{x}_i, \mathbf{x}_{\omega_m})]^{\top}$
(2) If coherence condition violated:	$\max_{k=1,\ldots,m} \lvert\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k})\rvert < \nu$
Increment the model order	$m = m + 1$, $\mathbf{x}_{\omega_m} = \mathbf{x}_i$, $\boldsymbol{\alpha}_{i-1} = [\boldsymbol{\alpha}_{i-1}^{\top} 0]^{\top}$
(3) Update coefficients	$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1} + \frac{\rho}{\|\boldsymbol{\kappa}_i\|^2} \boldsymbol{\kappa}_i (d_i - \boldsymbol{\kappa}_i^{\top} \boldsymbol{\alpha}_{i-1})$

Thus, the coherence criterion (11) is equivalent to a distance criterion in $\mathcal{H}$ where kernel functions are discarded if they are too close to those already in the model. Distance criteria are relevant within the context of sensor networks since they can be related to signal strength loss [10]. We shall discuss this property further at the end of the next section when we study the optimal selection of sensors.

3. Distributed Learning Algorithm

Let $\psi(\cdot) = \sum_{k=1}^{m} \alpha_k \kappa(\mathbf{x}_{\omega_k}, \cdot)$ be the mth order model where the kernels $\kappa(\mathbf{x}_{\omega_k}, \cdot)$ form a ν-coherent dictionary determined under the rule (11). In accordance with the least-squares problem (3), the m-dimensional coefficient vector $\boldsymbol{\alpha}^*$ satisfies

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{d} - \mathbf{H}\boldsymbol{\alpha}\|^2 + \eta \boldsymbol{\alpha}^{\top} \mathbf{K}_{\omega} \boldsymbol{\alpha}, \tag{14}$$

where $\mathbf{H}$ is the n-by-m matrix with (i, j)th entry $\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_j})$, and $\mathbf{K}_{\omega}$ is the m-by-m matrix with (i, j)th entry $\kappa(\mathbf{x}_{\omega_i}, \mathbf{x}_{\omega_j})$. The solution $\boldsymbol{\alpha}^*$ is obtained as follows:

$$\boldsymbol{\alpha}^* = \left(\mathbf{H}^{\top}\mathbf{H} + \eta \mathbf{K}_{\omega}\right)^{-1} \mathbf{H}^{\top} \mathbf{d}, \tag{15}$$

which requires $\mathcal{O}(m^3)$ operations as compared to $\mathcal{O}(n^3)$, with $m \ll n$, for the optimal solution given by (7). We shall now cut down the computational cost further by using a distributed algorithm in which each sensor node updates the coefficient vector.

3.1. Recursive Parameter Updating. To solve problem (15) recursively, we consider an optimization algorithm based on the principle of minimal disturbance, as studied in our paper [25]. Sensor i computes $\boldsymbol{\alpha}_i$ from $\boldsymbol{\alpha}_{i-1}$ received from sensor $i-1$ by minimizing the norm between both coefficient vectors under the constraint $\psi(\mathbf{x}_i) = d_i$. The optimization problem solved at sensor i is

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_{i-1} - \boldsymbol{\alpha}_i\|^2 \\ &\text{subject to } \boldsymbol{\kappa}_i^{\top}\, \boldsymbol{\alpha}_i = d_i, \end{aligned} \tag{16}$$

where $\boldsymbol{\kappa}_i$ is a m-dimensional column vector whose kth entry is $\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k})$. The model order control using (11) requires different measures for each of the two alternatives described next.

$\max_{k=1,\ldots,m} |\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k})| > \nu$. Sensor i is close to one of the previously selected sensors $\omega_1, \ldots, \omega_m$ in the sense of the norm in $\mathcal{H}$. Thus, the kernel function $\kappa(\mathbf{x}_i, \cdot)$ does not need to be inserted into the model, whose order remains unchanged. Only the coefficient vector needs to be updated. The solution to (16) can be obtained by minimizing the Lagrangian function

$$J(\boldsymbol{\alpha}_i, \lambda) = \|\boldsymbol{\alpha}_{i-1} - \boldsymbol{\alpha}_i\|^2 + \lambda(d_i - \boldsymbol{\kappa}_i^\top \boldsymbol{\alpha}_i), \tag{17}$$

where λ is the Lagrange multiplier. Differentiating this expression with respect to both $\boldsymbol{\alpha}_i$ and λ, and setting the derivatives to zero, we get the following equations

$$2(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_{i-1}) = \lambda \boldsymbol{\kappa}_i, \tag{18}$$

$$\boldsymbol{\kappa}_i^\top \boldsymbol{\alpha}_i = d_i. \tag{19}$$

Assuming that $\boldsymbol{\kappa}_i^\top \boldsymbol{\kappa}_i$ is nonzero, these equations yield

$$\lambda = 2\left(\boldsymbol{\kappa}_i^\top \boldsymbol{\kappa}_i\right)^{-1}(d_i - \boldsymbol{\kappa}_i^\top \boldsymbol{\alpha}_{i-1}). \tag{20}$$

Substituting the expression for λ into (18) leads to the following recursion

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1} + \frac{\rho}{\|\boldsymbol{\kappa}_i\|^2}\, \boldsymbol{\kappa}_i(d_i - \boldsymbol{\kappa}_i^\top \boldsymbol{\alpha}_{i-1}), \tag{21}$$

where we have introduced the step-size parameter ρ in order to control the convergence rate of the algorithm.

$\max_{k=1,\ldots,m} |\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k})| \leq \nu$. The topology defined by sensors $\omega_1, \ldots, \omega_m$ does not cover the region monitored by sensor i. The kernel function $\kappa(\mathbf{x}_i, \cdot)$ is then inserted into the model, and will henceforth be denoted by $\kappa(\mathbf{x}_{\omega_{m+1}}, \cdot)$. Now we have

$$\psi(\cdot) = \sum_{k=1}^{m+1} \alpha_k\, \kappa(\mathbf{x}_{\omega_k}, \cdot). \tag{22}$$

To accommodate the new entry α_{m+1}, we modify the optimization problem (16) as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_i} \left|\left|\boldsymbol{\alpha}_{i-1} - \boldsymbol{\alpha}_{i,[1:m]}\right|\right|^2 + \alpha_{m+1}^2, \\ &\text{subject to } \boldsymbol{\kappa}_i^\top\, \boldsymbol{\alpha}_i = d_i, \end{aligned} \tag{23}$$

where the subscript$_{[1:m]}$ denotes the first m elements of $\boldsymbol{\alpha}_i$. Note that $\boldsymbol{\kappa}_i$ now has one more entry, $\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_{m+1}})$. Writing the Lagrangian and setting to zero its derivatives with respect to $\boldsymbol{\alpha}_i$ and λ, we get the following updating rule

$$\boldsymbol{\alpha}_i = \begin{bmatrix} \boldsymbol{\alpha}_{i-1} \\ 0 \end{bmatrix} + \frac{\rho}{\|\boldsymbol{\kappa}_i\|^2} \boldsymbol{\kappa}_i \left(d_i - \boldsymbol{\kappa}_i^\top \begin{bmatrix} \boldsymbol{\alpha}_{i-1} \\ 0 \end{bmatrix}\right). \tag{24}$$

The form of recursions (21)–(24) is that of the kernel-based normalized LMS algorithm with order-update mechanism. The pseudocode of the algorithm is summarized in Table 1.

3.2. Algorithm and Remarks. We now illustrate the proposed approach. We shall address the problem of optimally selecting the sensors ω_k in the next subsection. Consider the network schematically shown in Figure 2. Here, each sensor is represented by a node, and communications between sensors are indicated by one-directional arrows. The process is initialized with sensor 1, that is, we set $\omega_1 = 1$ and $m = 1$. Let us suppose that sensors 2 and 3 belong to the neighborhood of sensor 1 with respect to criterion (11). As illustrated in Figure 2, this means that (11) is not satisfied for $k = 1$ and $i = 2, 3$. Thus, the model order remains unaltered when the algorithm processes the information at nodes 2 and 3. The coefficient vector $\boldsymbol{\alpha}_i$ is updated using rule (21) for $i = 2, 3$. Sensor-to-sensor communications transmit the locations of sensors that contribute to the model, here $\mathbf{x}_1$, and the updated parameter vector. As information propagates through the network, it may be transmitted to a sensor which satisfies criterion (11). This is the case of sensor 4, which is then considered to be outside the area covered by contributing sensors. Consequently, the model order m is increased by one at sensor 4 and the coefficient vector $\boldsymbol{\alpha}_4$ is updated using (24). Next, the sensor locations $[\mathbf{x}_1 \mathbf{x}_4]$ and the parameter vector $\boldsymbol{\alpha}_4$ are sent to sensor 5, and so on.

Updating cycles can be repeated to refine the model or to track time-varying systems. (Though beyond the scope of this paper, one may assume a time-evolution kernel-based model in the spirit of [4] where the authors fit a cubic polynomial to the temporal measurements of each sensor.)

For a network with a fixed sensor spatial distribution, the coefficient vectors $\boldsymbol{\alpha}_i$ tend to be adapted using (21) with a fixed-order m, after a transient period during which rules (21) and (24) are both used.

Note that different approaches may be used to derive the recursive parameter updating equation. The wide available literature on adaptive filtering methods [26, 27] can be used to derive different kernel-based adaptive algorithms that may have desirable properties for solving specific problems [23]. For instance, specific strategies may be used to tune the step-size parameter ρ in (21) and (24) for a better trade-off between convergence speed and steady-state performance. Note also that regularization is usually unnecessary in (21) and (24) for adequate values of ν. (Regularization would be implemented in (21) and (24) by using a step-size of the form $\rho/(\|\boldsymbol{\kappa}_i\|^2 + \eta)$, where η is the regularization coefficient.) If sensor i is one of the m model-contributing sensors, then $\kappa(\mathbf{x}_i, \mathbf{x}_i)$ is an entry of vector $\boldsymbol{\kappa}_i$. Assuming again without loss of generality that κ is a unit-norm kernel, this yields $\|\boldsymbol{\kappa}_i\| \geq 1$. Otherwise, sensor i does not satisfy criterion (11). This implies that there exists at least one index k such that $\kappa(\mathbf{x}_i, \mathbf{x}_{\omega_k}) > \nu$, and thus $\|\boldsymbol{\kappa}_i\| > \nu$.

3.3. On the Optimal Selection of Sensors ω_k. In order to design an efficient sensor network and to perform a proper dimensioning of its components, it is crucial that the order m of the model be as small as possible. So far, we have considered a simple heuristic consisting of visiting the sensor nodes as they are encountered, and selecting on-the-fly with criterion (11) those to include in the model. We

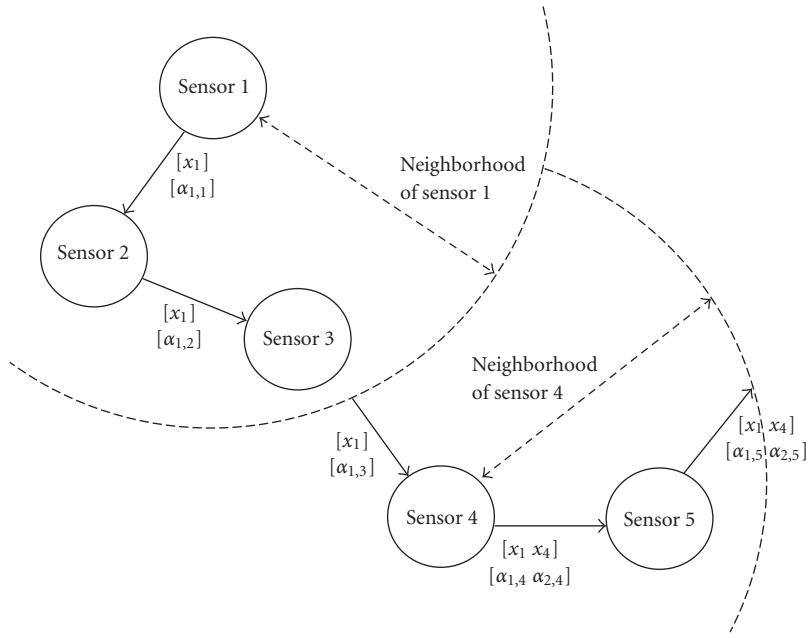


Figure 2: Illustration of the distributed learning algorithm.

shall now formalize this selection as a minimum set cover combinatorial optimization problem.

We consider the finite set $\mathcal{H}_n = \{\kappa(\mathbf{x}_1, \cdot), \ldots, \kappa(\mathbf{x}_n, \cdot)\}$ of kernel functions, and the family of disks of radius ν centered at each $\kappa(\mathbf{x}_k, \cdot)$. Within this framework, a set cover is a collection of some of these disks whose union is $\mathcal{H}_n$. Note that we have denoted by $\mathcal{D}_m$ the set containing the $\kappa(\mathbf{x}_{\omega_k}, \cdot)$'s, with $k = 1, \ldots, m$. In the set covering optimization problem, the question is to find a collection $\mathcal{D}_m$ with minimum cardinality. This problem is known to be NP-hard. The linear programming relaxation of this 0-1 integer program has been considered by numerous authors, starting with the seminal work [28]. Greedy algorithms have also received attention as they provide good or near-optimal solutions in a reasonable time [29]. Greedy algorithms make the locally optimum choice at each iteration, without regard for its implications on future stages. To insure stochastic behavior, randomized greedy algorithms have been proposed [30, 31]. They often generate better solutions than the pure greedy ones. To improve the solution quality, sophisticated heuristics such as simulated annealing [32], genetic algorithms [33], and neural networks [34] introduce randomness in a systematic manner.

Consider, for instance, the use of the basic greedy algorithm to determine $\mathcal{D}_m$. The greedy algorithm for set covering chooses, at each stage, the set which contains the largest number of uncovered elements. It can be shown that this algorithm achieves an approximation ratio of the optimum equal to $H(p) = \sum_{k=1}^{p} 1/k$, where p is the size of the largest set of the cover. To illustrate the effectiveness of this approach for sensor selection, and to compare it with the on-the-fly method discussed previously, 100 sensors were randomly deployed over a 1.6-by-1.6 square area. The variation of the number of selected sensor nodes as a function of the coherence threshold was examined. To provide numerical results independent of the kernel form, and thus to simplify the presentation, criterion (11) was replaced by the following. (In the case where κ is a strictly decreasing function, criterion (11) can be rewritten as $\|\mathbf{x}_i - \mathbf{x}_{\omega_k}\| < \nu_x$ with $\nu_x = \kappa^{-1}(1 - \nu/2)$.) For instance, with the Gaussian kernel, we have $\nu_x = \sqrt{-2\beta_0 \ln(1 - \nu/2)}$

$$\max_{k=1,\ldots,m} \|\mathbf{x}_i - \mathbf{x}_{\omega_k}\| \leq \nu_x. \tag{25}$$

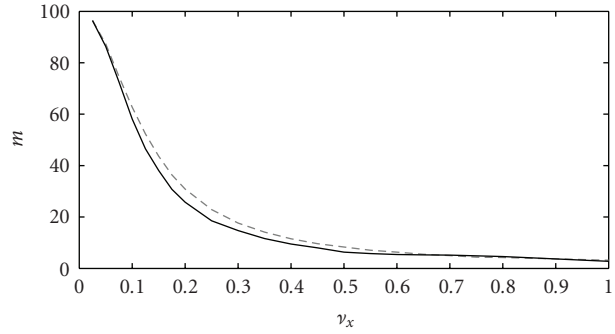


Figure 3: Number of sensor nodes selected by the greedy (solid) and the on-the-fly (dashed) algorithms as a function of the coherence threshold.

The results are reported in Figure 3, and illustrative examples are shown in Figure 4. These examples indicate that the greedy algorithm, which is based on centralized computing, performs only slightly better than the on-the-fly method. Moreover, it can be observed that m tends rapidly to moderate values in both cases. Our experience has shown that the application of either algorithm leads to a model

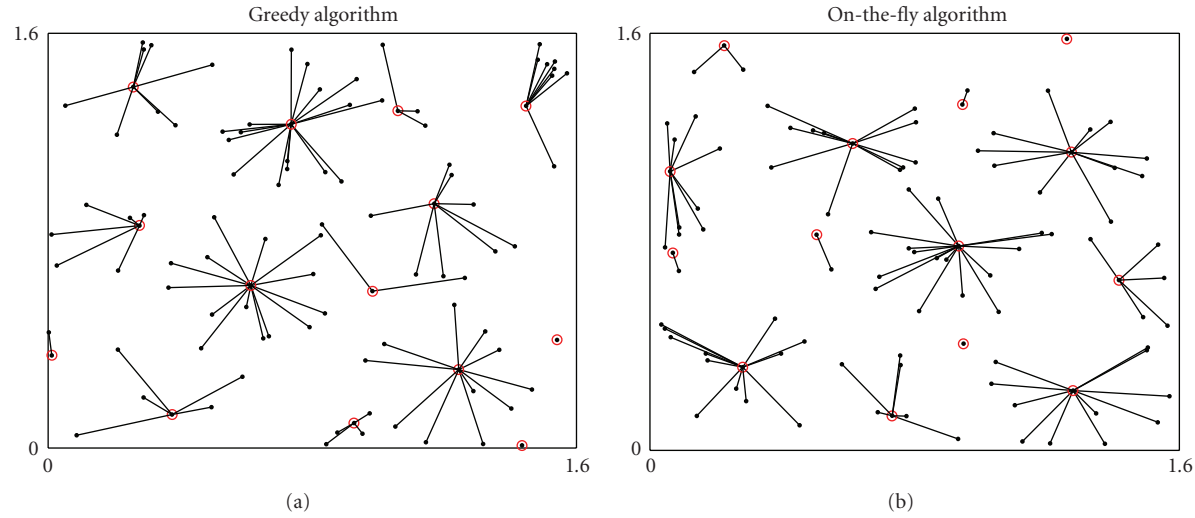


FIGURE 4: Cluster heads (red dots) and slaves (black dots) obtained for $\nu_x = 0.30$ using the greedy and the on-the-fly algorithms. The numbers of cluster heads obtained over 100 sensor nodes were equal to 14 and 18, respectively.

order m which is at least one order of magnitude smaller than the number of sensors n. This property will be illustrated in Section V, where the results obtained using the elementary decentralized on-the-fly algorithm indicate that there is room for further improvement of the proposed approach.

4. Resource Requirements

Algorithms designed to be deployed in sensor networks must be evaluated regarding their requirements for energy consumption, computational complexity, memory allocation, and communications. Most of these requirements are interrelated, and thus cannot be analyzed independently. In this section, we provide a brief discussion of several aspects regarding such requirements for the proposed algorithm, assuming for simplicity that each sensor requires similar resources to receive a message from the previous sensor, update its content, and send it to the next one.

Energy-Accuracy Trade-Off. the proposed solution allows for a trade-off between energy consumption and model accuracy. This trade-off can be adjusted according to the application requirements. Consider the case of a large neighborhood threshold ν. Then, applying rule (11), each sensor will have many neighbors and the resulting model order will be low. This will result in low computational cost and power consumption for communication between sensors, at the price of a coarse approximation. On the other hand, a small value for ν will result in a large model order. This will lead to a small approximation error, at the price of high computational load for updating the model at each sensor, and high power requirements for communication. This is the well-known energy-accuracy dilemma.

Localization. as each node needs to know its location, a preprocessing stage for sensor autolocalization is often required. The available techniques for this purpose can be grouped into centralized and decentralized ones. See, for example, [10, 35–37] and references therein. The former requires the transmission of ranging information, such as distance or received signal strength measurements, from sensors to a fusion center. The latter makes each sensor location-aware using information gathered from its neighbors. The decentralized approach is more energy-efficient, in particular for large-scale sensor networks, and should be preferred over the centralized one. Note that the model (9) locally requires the knowledge of only m out of the n sensor locations, with $m \ll n$.

Computational Complexity. the m-dimensional parameter updating presented in this paper uses, at each sensor node, an LMS-based adaptive algorithm. LMS-based algorithms are very popular in industrial applications, mainly because of their low complexity—$\mathcal{O}(m)$ operations per updating cycle and sensor—and their numerical robustness [26].

Memory Storage. each sensor node must store its coordinates, the coherence threshold ν, the step-size parameter ρ, and the parameters of the kernel. Unlike conventional techniques, the proposed algorithm does not require storing information about local neighborhoods. Each sensor node needs to know is if it is a neighbor of the model-contributing sensors. This is determined by evaluating rule (11) using the locations $\mathbf{x}_{\omega_k}$ transmitted by the last active node.

Energy Consumption. communications account for most of the energy consumption in wireless sensor networks. The energy spent in communication is often dramatically greater than the energy consumption incurred by in-sensor computations, although the latter is difficult to estimate accurately. Consider, for instance, the energy dissipation model introduced in [7]. According to this model, the energy

required to transmit one bit between two sensors ℓ meters apart is given by $E_{\text{amp}}\ell^2 + E_{\text{elec}}$, where E_{elec} denotes the electronic energy, and E_{amp} is the amplifier energy. E_{elec} depends on the signal processing required, and E_{amp} depends on the acceptable bit-error rate. The energy cost incurred by the reception of one bit can be modeled as well by E_{elec}. Therefore, the energy dissipation is quadratic in the routing distance, and linear in the number of bits sent. The proposed algorithm transmits information between neighboring sensors and requires the transmission of a small amount of information only.

Evaluation. in a postprocessing stage of the distributed learning algorithm, the model is used to estimate the investigated spatial distribution at given locations. From (9), this requires m evaluations of the kernel function, m multiplications with the weighting coefficients, and m additions. This reinforces the importance of a reduced model order m, as provided by the proposed algorithm.

5. Simulations

The emerging world of wireless sensor networks suffers from lack of real system deployments and available data experiments. Researchers often evaluate their algorithms and protocols with model-driven data [38]. Here, we consider a classical application of estimating a temperature field simulated using a partial differential equation solver. Before proceeding, let us describe the experimental setup. Heat propagation in an isotropic and homogeneous medium can be modeled by the partial differential equation

$$\mu C \frac{\partial T(\mathbf{x},t)}{\partial t} - k\nabla_{\mathbf{x}}^2 T(\mathbf{x},t) = Q(\mathbf{x},t) + h(T(\mathbf{x},t) - T_{\text{ext}}), \tag{26}$$

where $T(\mathbf{x},t)$ is the temperature as a function of location and time, μ and C the density and the heat capacity of the medium, k the coefficient of heat conduction, $Q(\mathbf{x},t)$ the heat sources, h the convective heat transfer coefficient, and T_{ext} the external temperature. In the above equation, $\nabla_{\mathbf{x}}^2$ denotes the Laplace spatial operator. Two sets of experiments were conducted. In the first experimental setup, we considered the problem of estimating a static spatial temperature distribution, and studied the influence of different tuning parameters on the convergence of the algorithm. In the second experimental setup, we studied the problem of monitoring the evolution of the temperature over time.

5.1. Estimation of a Static Field. As a first benchmark problem, we reduce the propagation equation (26) to the following partial derivative equation of parabolic type

$$-\nabla_{\mathbf{x}}^2 T(\mathbf{x}) = Q(\mathbf{x}) + T(\mathbf{x}). \tag{27}$$

The region of interest is a 1.6-by-1.6 square area with open boundaries and two circular heat sources dissipating 20 W. In order to estimate the spatial distribution of temperature at any location, 100 sensors were randomly deployed according to a uniform distribution. The desired outputs $T(\mathbf{x})$, generated by using the Matlab PDE solver, were corrupted by a measurement noise sampled from a zero-mean Gaussian distribution with standard deviation σ equal to 0.01 at first, and next equal to 0.1. This led to signal-to-noise ratios, defined as the ratio of the powers of $T(\mathbf{x})$ and the additive noise, of 9.7 dB and 25 dB, respectively. These data were used to estimate a nonlinear model of the form $T_n = \psi(\mathbf{x}_n)$ based on the Gaussian kernel. Preliminary experiments were conducted as explained below to determine all the adjustable parameters, that is, the kernel bandwidth β_0 and the step-size ρ. To facilitate comparison between different settings, we fixed the threshold ν_x introduced in (25) rather than the coherence ν presented in (11). The algorithm was then evaluated on several independent test signals. This led to the learning curves depicted in Figure 5, and to the performance reported in Table 2. An estimation of the temperature field is provided in Figure 6.

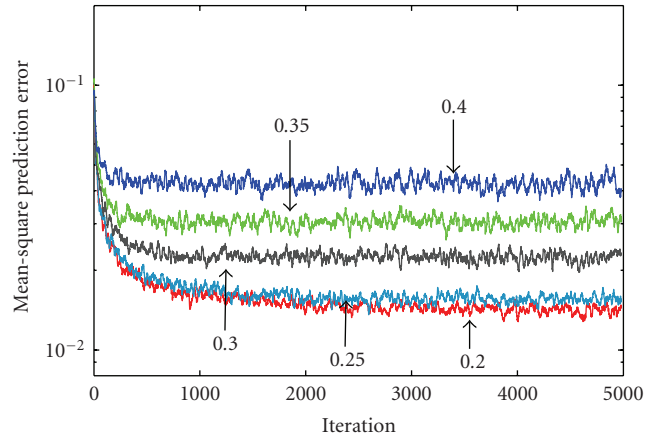


Figure 5: Learning curves for ν_x varying from 0.20 to 0.40, obtained by averaging over 200 experiments.

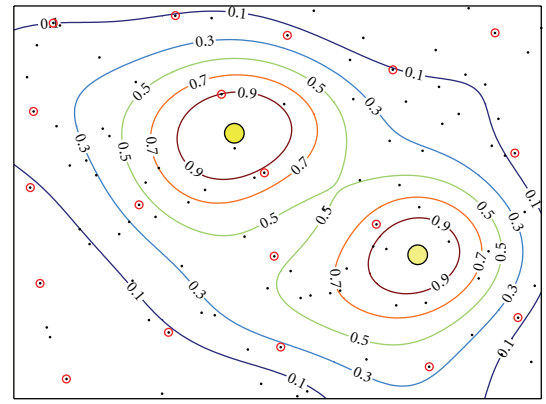


Figure 6: Spatial distribution of temperature estimated using 100 sensors. Parameters were set as follows: $\rho = 0.3$, $\beta_0 = 0.24$, $\nu_x = 0.30$, $\sigma = 0.1$. The resulting model order was $m = 19$. The heat sources are represented by two yellow discs. The sensors are indicated by black dots. The latter are red-circled in the case of cluster heads.

Table 2: Parameter settings, performance and model order as a function of the measurement noise level.

Parameter settings			NMSE		
ρ	β_0	ν_x	$\sigma = 0.01$	$\sigma = 0.1$	$\overline{m}$
0.3	0.24	0.20	0.015	0.081	31.8
		0.25	0.025	0.091	23.4
		0.30	0.064	0.130	17.8
		0.35	0.117	0.171	14.2
		0.40	0.176	0.248	11.7

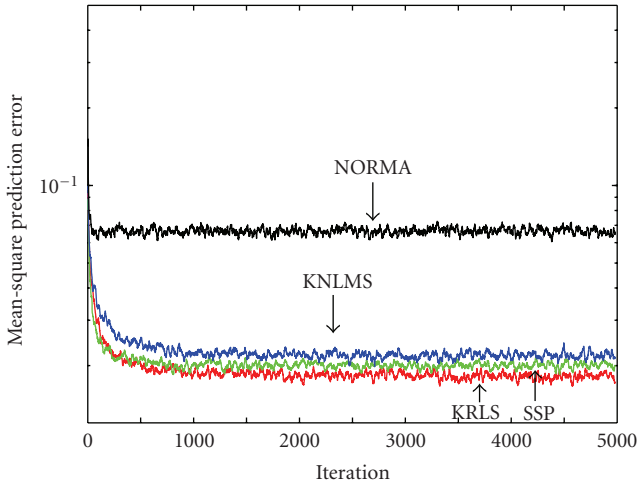


Figure 7: Learning curves for KNLMS, NORMA, SSP, and KRLS obtained by averaging over 200 experiments.

The preliminary experiments were conducted on sequences of 5000 noisy samples, which were obtained by visiting the 100 sensor nodes along a random path. These data were used to determine β_0 and ρ, for given ν_x. Performance was measured in steady-state using the mean-square prediction error $1/1000 \sum_{n=4001}^{5000} (T_n - \psi_{n-1}(\mathbf{x}_n))^2$ over the last 1000 samples of each sequence and averaged over 40 independent trials. The threshold ν_x was varied from 0.20 to 0.40 in increments of 0.05. Given ν_x, the best performing step-size parameter ρ and kernel bandwidth β_0 were determined by grid search over the intervals $(0.05 \leq \rho \leq 0.7) \times (0.14 \leq \beta_0 \leq 0.26)$ with increments 0.05 and 0.02, respectively. A satisfactory compromise between convergence speed and accuracy was reached with $\beta_0 = 0.24$ and $\rho = 0.3$. The algorithm was tested over two hundred 5000-sample independent sequences, with the parameter settings obtained as described above and specified in Table 2. This led to the ensemble-average learning curves shown in Figure 5. Steady-state performance was measured by the normalized mean-square prediction error over the last 1000 samples, defined as follows:

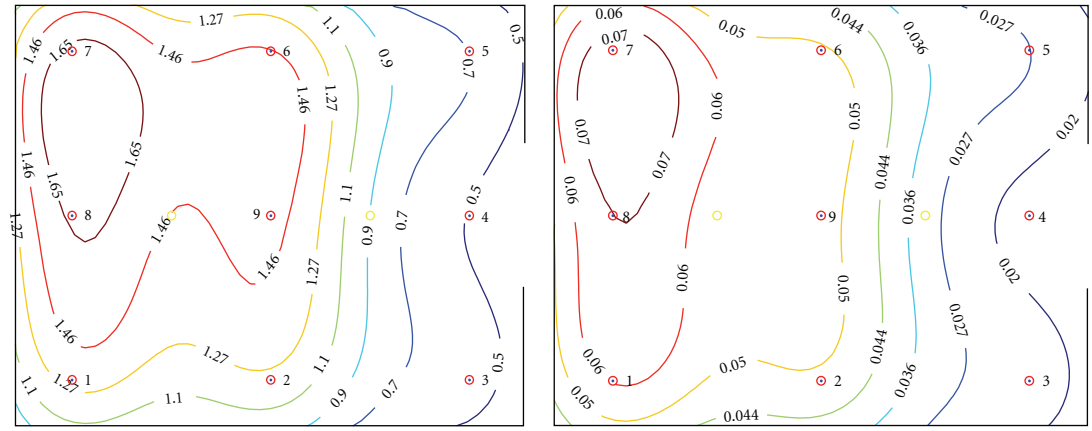
$$\text{NMSE} = E\left\{ \frac{\sum_{n=4001}^{5000} \left(T_n - \psi_{n-1}(\mathbf{x}_n)\right)^2}{\sum_{n=4001}^{5000} T_n^2} \right\}, \tag{28}$$

where the expectation was approximated by averaging over the ensemble. Table 2 also reports the sample mean values $\overline{m}$ for the model order m over the two hundred test sequences. It indicates that the prediction error decreased as $\overline{m}$ increased and ν_x decreased. Note that satisfactory levels of performance were reached with small model orders.

For comparison purposes, state-of-the-art kernel-based methods for online prediction of time series were also considered: NORMA [39], Sparse Sequential Projection (SSP) [40], and KRLS [22]. As the KNLMS algorithm, NORMA performs stochastic gradient descent on RKHS. The order of the kernel expansion is fixed a priori since it uses the m most recent kernel functions as a dictionary. NORMA requires $\mathcal{O}(m)$ operations per iteration. SSP method also starts with stochastic gradient descent to calculate the a posteriori estimate. The resulting $(m + 1)$-order kernel expansion is then projected onto the subspace spanned by the m kernel functions of the dictionary, and the projection error is compared to a threshold in order to evaluate whether the contribution of the $(m + 1)$th candidate kernel function is significant enough. If not, the projection is used as the a posteriori estimate. In the spirit of the sparsification rule (10), this test requires $\mathcal{O}(m^2)$ operations per iteration when implemented recursively. KRLS is a RLS-type algorithm with, in [22], an order-update process controlled by the condition (10). Its computational complexity is also $\mathcal{O}(m^2)$ operations per iteration. Table 3 reports a comparison of the estimated computational costs per iteration for each algorithm, in the most usual situation where no order increase is performed. These results are expressed for real-valued data in terms of the number of real multiplications and real additions.

The temperature distribution $T(\mathbf{x})$ considered previously, corrupted by a zero-mean white Gaussian noise with standard deviation σ equal to 0.1, was used to estimate a nonlinear model of the form $T_n = \psi(\mathbf{x}_n)$ based on the Gaussian kernel. The same initialization process used for KNLMS was followed to initialize and test NORMA, SSP and KRLS. This means that preliminary experiments were conducted on 40 independent 5000-sample sequences to perform explicit grid search over parameter spaces and, following the notations used in [22, 39, 40], to select the best settings reported in Table 3. For an unambiguous comparison of these algorithms, note that their sparsification rules were individually hand-tuned, via appropriate threshold selection, to provide models with approximately the same order m. In addition, the Gaussian kernel bandwidth β_0 was set to 0.24 for all the algorithms. Each approach was tested over two-hundred 5000-sample sequences, which led to the normalized mean-square prediction errors displayed in Table 3. As shown in Figure 7, the algorithms with quadratic complexity performed better than the other two, with only a small advantage of SSP over KNLMS. Obviously, this must be balanced with the large increase in computational cost. This experiment also highlights that KNLMS significantly outperformed the other algorithm with linear complexity, namely, NORMA, which clearly demonstrates the effectiveness of our approach.

5.2. Tracking of a Dynamic Field. As a second application, we consider the problem of heat propagation governed by



(a) (b)

Figure 8: Spatial distribution of temperature estimated at time instants 10 (a) and 20 (b), when the heat sources are turned off and on.

Table 3: Estimated computational cost per iteration, experimental setup and performance.

Algorithm	×	+	Parameter settings	$\overline{m}$	NMSE
NORMA	$2m$	m	$\lambda = 0.3, \eta = 0.7$	18	0.3514
KNLMS	$3m + 1$	$3m$	$\nu_x = 0.3, \eta = 0.3$	18.17	0.1243
SSP	$3m^2 + 6m + 1$	$3m^2 + m - 1$	$\kappa = 0.08$	18.03	0.1164
KRLS	$4m^2 + 4m$	$4m^2 + 4m + 1$	$\nu = 0.68$	18.13	0.1041

equation (26) in a partially bounded conducting medium. As can be seen in Figure 8, the region of interest is a 2-by-3 rectangular area with two heat sources that dissipate 2000 W when turned on. This area is surrounded by a boundary layer with low conductance coefficient, except on the right side where an opening exists. The parameters used in the experimental setup considered below include

$$\begin{aligned} &\text{rectangular area:} \\ &(\mu C)_r = 1 \quad k_r = 10 \quad h_r = 0, \\ &\text{boundary layer:} \\ &(\mu C)_b = 1 \quad k_b = 0.1 \quad h_b = 0. \end{aligned} \tag{29}$$

The heat sources were simultaneously turned on or off over periods of 10 time steps. In order to estimate the spatial distribution of temperature at any location, and track its evolution over time, 9 sensors were deployed in a grid. The desired outputs $T(\mathbf{x}, t)$ were generated using the Matlab PDE solver. They were corrupted by an additive zero-mean white Gaussian noise with standard deviation σ equal to 0.08, corresponding to a signal-to-noise ratio of 10 dB. These data were used to estimate a nonlinear model, based on the Gaussian kernel, that predicts temperature as a function of location and time.

Preliminary experiments were conducted to determine the adjustable parameters of our algorithm, using 100 independent sequences of 360 noisy samples. Each sequence was obtained by collecting, simultaneously, the 9 sensor readings over 2 on-off source periods. Performance was measured with mean-square prediction error, which was averaged over the 100 sequences. Due to the small number of available sensors, no condition on coherence was imposed via ν or ν_x. This led to models of order $m = n = 9$. The best performing kernel bandwidth β_0 and step-size parameter ρ were determined by grid search over the interval $(0.3 \leq \beta_0 \leq 0.7) \times (0.5 \leq \rho \leq 2.0)$ with increment 0.05 for both β_0 and ρ. A satisfactory compromise between convergence speed and accuracy was reached by setting β_0 to 0.5, and ρ to 1.55.

The algorithm was tested over two hundred 360-sample sequences prepared as above. This led to the predicted temperature curves depicted in Figure 9, which demonstrate the ability of our technique to track local changes. Figure 8 provides two snapshots of the spatial distribution of temperature, at time instants 10 and 20, when the heat sources are turned off and on. We can observe the flow of heat from inside the container to the outside, through the opening in the right side.

6. Conclusion

Over the last ten years or so, there has been an explosion of activity in the field of learning algorithms utilizing reproducing kernels, most notably in the field of classification and regression. The use of kernels is an attractive computational shortcut to create nonlinear versions of conventional linear algorithms. In this paper, we have demonstrated the versatility and utility of this family of methods to develop a nonlinear adaptive algorithm for time series prediction

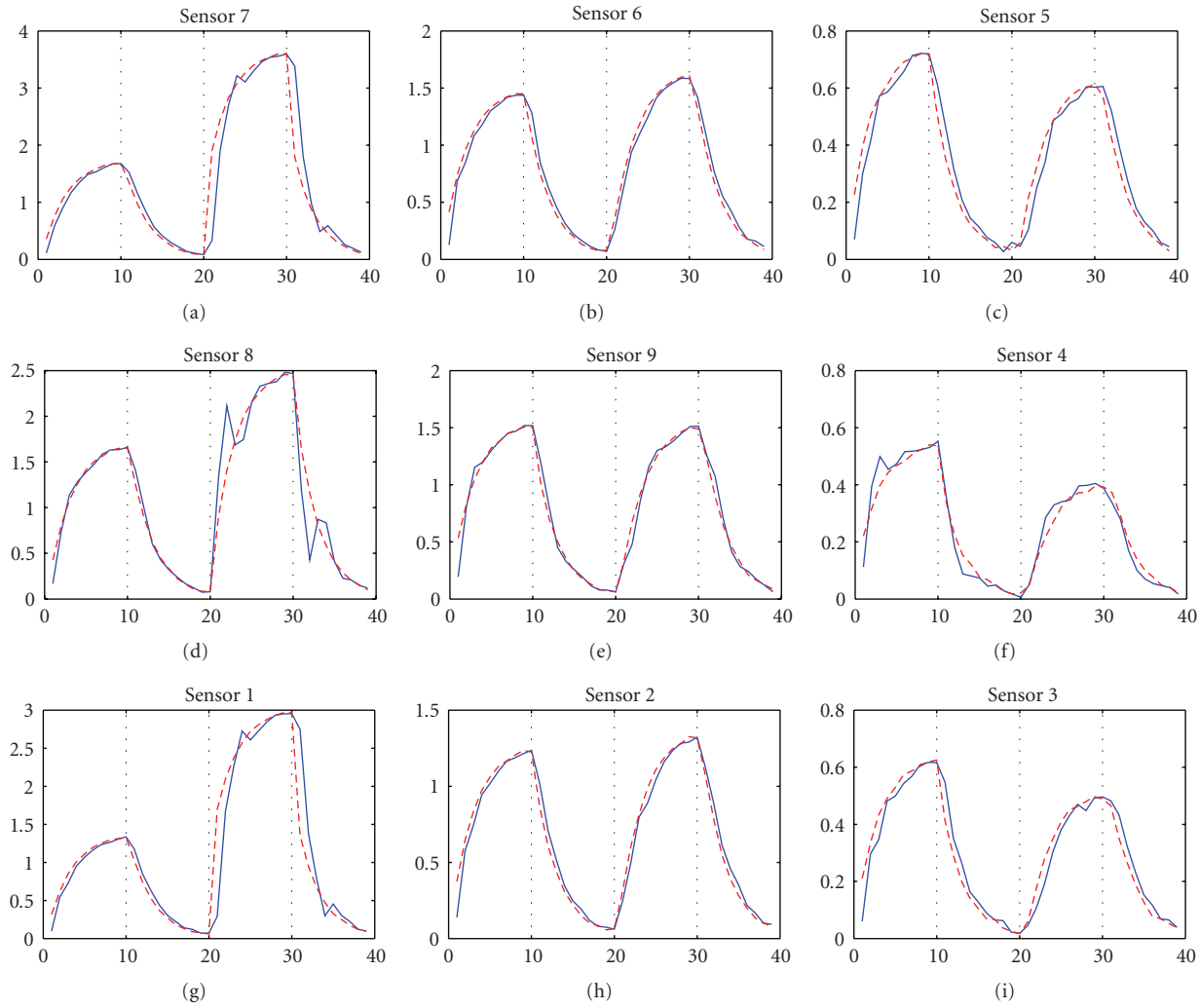


Figure 9: Evolution of the predicted (solid blue) and measured (dashed red) temperatures by each sensor. Both heat sources were turned on over the intervals $[0, 10]$ and $[20, 30]$, and turned off over $[10, 20]$ and $[30, 40]$. Sensor locations can be found in Figure 8.

in sensor networks. A common characteristic in kernel-based methods is that they deal with models whose order equals the size of the training set, making them unsuitable for online applications. Therefore, it was essential to first propose a mechanism for controlling the increase in the model order as new input data become available. This led us to consider the coherence criterion. We incorporated it into a kernel-based normalized LMS algorithm with order-update mechanism, which is a notable contribution to our study. Our approach has demonstrated good performance during experiments.

Perspectives include the derivation of new sparsification criteria also based on sensor readings rather than being limited to sensor locations. This would certainly result in better performance on the prediction of dynamic fields. Online optimization of this criterion by adding or removing kernel functions from the dictionary also seems interesting. Finally, in a broader perspective, controlling the movement of sensor nodes, when allowed by the application, to achieve improved estimation accuracy appears as a very promising subject for research.

References

[1] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 20–27, ACM, New York, NY, USA, April 2004.

[2] S.-H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz, "The value of clustering in distributed estimation for sensor networks," in *Proceedings of the International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 2, pp. 969–974, June 2005.

[3] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[4] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Proceedings of the 3rd International*

Symposium on Information Processing in Sensor Networks (IPSN '04), pp. 1–10, ACM, New York, NY, USA, April 2004.

[5] J. B. Predd, S. R. Kulkarni, and H. Vincent Poor, "Regression in sensor networks: training distributively with alternating projections," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XV*, vol. 5910 of *Proceedings of SPIE*, pp. 1–15, San Diego, Calif, USA, 2005.

[6] J. B. Predd, S. R. Kulkarni, and H. Vincent Poor, "Distributed kernel regression: an algorithm for training collaboratively," in *Proceedings of the Information Theory Workshop*, 2006.

[7] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[8] S. Lindsey and C. S. Raghavendra, "Pegasis: power-efficient gathering in sensor information systems," in *Proceedings of the Aerospace Conference*, vol. 3, pp. 1125–1130, 2002.

[9] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, NY, USA, 1998.

[10] X. Nguyen, M. I. Jordan, and B. Sinopoli, "A kernel-based learning approach to ad hoc sensor network localization," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 134–152, 2005.

[11] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Nonparametric decentralized detection using kernel methods," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4053–4066, 2005.

[12] M. C. Vuran and I. F. Akyildiz, "Spatial correlation-based collaborative medium access control in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 316–329, 2006.

[13] A. Jindal and K. Psounis, "Modeling spatially correlated data in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 466–499, 2006.

[14] Z. Quan, W. J. Kaiser, and A. H. Sayed, "A spatial sampling scheme based on innovations diffusion in sensor networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 323–330, ACM, New York, NY, USA, April 2007.

[15] R. Herbrich, *Learning Kernel Classifiers. Theory and Algorithms*, The MIT Press, Cambridge, Mass, USA, 2002.

[16] G. Kimeldorf and G. Wahba, "Some results on Tchebycheffian spline functions," *Journal of Mathematical Analysis and Applications*, vol. 33, no. 1, pp. 82–95, 1971.

[17] B. Schölkopf, R. Herbrich, and R. Williamson, "A generalized representer theorem," Tech. Rep. NC2-TR-2000-81, Royal Holloway College, University of London, London, UK, 2000.

[18] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[19] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[20] G. Baudat and F. Anouar, "Kernel-based methods and function approximation," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, vol. 5, pp. 1244–1249, Washington, DC, USA, July 2001.

[21] L. Csató and M. Opper, "Sparse representation for gaussian process models," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13, pp. 444–450, The MIT Press, Cambridge, Mass, USA, 2001.

[22] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[23] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[24] P. Honeine, C. Richard, and J. C. Bermudez, "On-line nonlinear sparse approximation of functions," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 956–960, Nice, France, June 2007.

[25] P. Honeine, M. Essoloh, C. Richard, and H. Snoussi, "Distributed regression in sensor networks with a reduced-order Kernel model," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 112–116, New Orleans, LA, USA, December 2008.

[26] S. Haykin, *Adaptive Filtering Theory*, Prentice Hall, Upper Saddle River, NJ, USA, 4th edition, 2002.

[27] A. Sayed, *Fundamentals of Adaptive Filtering*, Wiley-IEEE, New York, NY, USA, 2003.

[28] L. Lovász, "On the ratio of optimal integral and fractional covers," *Discrete Mathematics*, vol. 13, no. 4, pp. 383–390, 1975.

[29] V. Chvátal, "A greedy heuristic for the set covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[30] F. J. Vasko and G. R. Wilson, "An efficient heuristic for large set covering problems," *Naval Research Logistics Quarterly*, vol. 31, no. 1, pp. 163–171, 1984.

[31] T. A. Féo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[32] L. W. Jacobs and M. J. Brusco, "Note: a local-search heuristic for large set-covering problems," *Naval Research Logistics*, vol. 42, no. 7, pp. 1129–1140, 1995.

[33] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.

[34] M. Aourid and B. Kaminska, "Neural networks for the set covering problem: an application to the test vector compaction," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 4645–4649, June 1994.

[35] N. Patwari and A. O. Hero I II, "Manifold learning algorithms for localization in wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 857–860, May 2004.

[36] J. Bachrach and C. Taylor, "Localization in sensor networks," in *Handbook of Sensor Networks*, I. Stojmenovic, Ed., 2005.

[37] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[38] Y. Yu, *Scalable, synthetic, sensor network data generation*, Ph.D. thesis, University of California, Los Angeles, Calif, USA, 2005.

[39] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[40] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison, "Function estimation in Hilbert space using sequential projections," in *Proceedings of the IFAC Conference on Intelligent Control Systems and Signal Processing*, pp. 113–118, 2003.