sensors

MDPI

Article

Learning Wireless Sensor Networks for Source Localization

S. Hamed Javadi [1,*], Hossein Moosaei [2] and Domenico Ciuonzo [3]

1 Department of Electrical Engineering, Faculty of Engineering, University of Bojnord, Bojnord 9453155111, Iran
2 Department of Mathematics, Faculty of Science, University of Bojnord, Bojnord 9453155111, Iran; hmoosaei@gmail.com
3 Department of Electrical Engineering and Information Technologies, University of Napoli "Federico II", 80100 Naples, Italy; domenico.ciuonzo@gmail.com
* Correspondence: h.javadi@ub.ac.ir; Tel.: +98-930-387-0916

Received: 30 December 2018; Accepted: 30 January 2019; Published: 2 February 2019

Abstract: Source localization and target tracking are among the most challenging problems in wireless sensor networks (WSN). Most of the state-of-the-art solutions are complicated and do not meet the processing and memory limitations of the existing low-cost sensor nodes. In this paper, we propose computationally-cheap solutions based on the support vector machine (SVM) and twin SVM (TWSVM) learning algorithms in which network nodes firstly detect the desired signal. Then, the network is trained to specify the nodes in the vicinity of the source (or target); hence, the region of event is detected. Finally, the centroid of the event region is considered as an estimation of the source location. The efficiency of the proposed methods is shown by simulations.

Keywords: Internet of Things; quadratic programming; region of event detection; source localization; support vector machine (SVM); target tracking; twin support vector machine (TWSVM); wireless sensor network (WSN)

1. Introduction

The applications of the networks of wireless smart motes with sensing, processing and communication capabilities are expanding. Among all the possible applications, these networks are applicable for preventing people from suffering health effects such as sleep disturbance [1,2], annoyance [3], cardiovascular effects [4], learning impairment [5,6], and hypertension ischemic heart disease [7]. Other examples include noise monitoring [8] for obtaining noise maps and related action plans [9].

These wireless sensor networks (WSN) are the basis of the emerging technology of Internet of Things (IoT) and specifically fit very well in surveillance applications [10,11], where detecting an event source in a region of interest (ROI) and tracking it are important challenges.

Tracking over WSNs is a challenging problem since it needs implementing statistical filters with cumbersome computations. These computationally heavy algorithms should be run by each network node with severe limitations of processing power, memory, and communication.

The nodes' estimations of the source track are transmitted to a central unit of network which is referred to as the "fusion center (FC)". FC adopts an appropriate fusion rule for obtaining a tuned estimation of the track. Some of the existing fusion rules are Independent likelihood pool (ILP) [12], Covariance intersection (CI) fusion [13], Information graph [14], Track-to-track fusion [15], and Consensus-based fusion (distributed MTT–DMTT) [16].

There are also source localization approaches based on information theoretic methods. In the approach presented in [17], a coarse estimation of the source location is obtained based on the data of specified nodes which are assigned as the anchor nodes. Then, a set of non-anchor nodes with maximum mutual information (MI) between the source location and their measurements are activated. The source is localized after several iterations. The complexity of the MI-based method grows exponentially in the network size. To alleviate this problem, another sensor selection scheme for source localization based on conditional posterior Cramer-Rao Lower Bound (PCRLB) has been proposed by [18,19].

In many applications, it is important to detect the region of event after detection of its occurrence. In other words, many applications need to divide ROI into event and non-event regions after the event occurrence is detected. For example, in fire detection in forests or poisonous gas leakage applications, it is crucial to detect the region of event during time in order to estimate the event expansion rate and schedule appropriate plans.

Due to severe bandwidth and energy limitations of battery-powered sensor nodes, they are usually programmed to send as little data as possible. The extreme case in event detection is implementation of distributed detection [20] where each node sends just one bit to FC indicating its decision about event occurrence. Apparently, the detection performance of nodes is not perfect and includes false alarms and misses of the event occurrence. However, if erroneous decisions of nodes can be corrected, the region of nodes with decision '1' (indicating event occurrence) would imply the event region. The goal of this paper is to correct both false alarms and misses of the network nodes using appropriate machine learning (ML) methods in order to estimate both event location and its region.

In any WSN, we encounter distributed agents gathering data for a specific task. Since the size of data collected by WSNs is usually enormous, the related problems and challenges can be handled by suitable ML methods. Instances of applying ML methods on different aspects of WSNs—such as node localization, channel allocation, and routing—have been reviewed in [21].

Support vector machines (SVMs) are powerful tools for classification and regression [22–24] which are based on the statistical learning theory. Compared with other ML methods such as artificial neural networks, SVMs have a better generalization guarantee. SVM divides the data samples of two classes by determining a specific hyperplane in the input space. This hyperplane maximizes the separation between the two classes by solving a quadratic programming problem (QPP) whose solution is globally optimum.

Inspired by the idea of SVM, twin support vector machine (TWSVM) for classification problems was proposed in [25]. TWSVM seeks two non-parallel proximal hyperplanes for classifying data sets. Unlike SVM, TWSVM solves two small SVM-type problems; i.e., TWSVM solves a pair of QPPs instead of a single complex QPP as in SVM. SVM and TWSVM also work effectively for classifying data sets which are not linearly separable by using the theory of kernel functions [26].

In this paper, two source localization methods are proposed based on distributed detection together with SVM and TWSVM algorithms. Our contributions presented in this paper are as follows:

- Distributed detection is implemented over a WSN in order to alleviate both computational and communication burdens (the two most important limitations of WSNs);
- By applying the SVM and TWSVM learning methods, the network becomes capable of detecting the nodes in vicinity of the desired event by whom the region of the event is detected;
- The event location is assumed to be the centroid of the event region. However, a correction method is provided for cases wherein the event occured at the edges of ROI;
- We show that our proposed methods—referred to as "Red-S" and "Red-T", respectively, since they perform the region of event detection (Red) by applying SVM and TWSVM, respectively—are not only good at source localization but also are capable of tracking a moving source.

The proposed algorithms are more general and practical than the existing learning methods such as the one in [27] where the region of event is detected via data exchange among neighbor nodes and the false alarm probability is restrictively assumed the same as the miss probability.

The remaining of this paper is organized as follows. The models and assumptions used throughout this manuscript are discussed in Section 2. Section 3 provides the required background knowledge. Estimation of the event location and region of event detection by using SVM and TWSVM are proposed in Section 4. The evaluation results is discussed in Section 5. Finally, the paper is concluded in Section 6.

Notations: Lower-case (resp. upper-case) bold letters denote vectors (resp. matrices), with a_i (resp. $A_{i,j}$) representing the ith element (resp. (i,j)th element) of $\boldsymbol{a}$ (resp. $\boldsymbol{A}$). In addition, $\boldsymbol{I}$ is used for denoting the identity matrix, $\mathbf{1}$ is a vector of all elements 1 with proper size, $\boldsymbol{a}^T$ (resp. $\boldsymbol{A}^T$) denotes the transpose of $\boldsymbol{a}$ (resp. $\boldsymbol{A}$), the 2-norm of vector $\boldsymbol{a}$ will be denoted by $\|\boldsymbol{a}\|$, and $\mathbb{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. In addition, $\mathcal{U}(a,b)$ indicates the uniform distribution between a and b. Finally, the symbol $\sim$ means "distributed as".

2. System Model and Assumptions

The system model shown in Figure 1 is adopted throughout this paper. Different parts of this model are discussed in the following subsections.

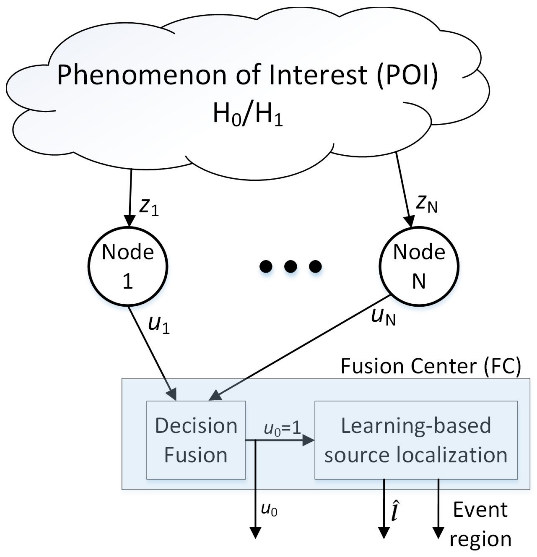

Figure 1. The network configuration studied in this paper. The network nodes observe a common phenomenon of interest and decide locally about event occurrence. Then, they send their decisions to FC where the final decision u_0 is taken. If $u_0 = 1$, FC gives an estimation of source location $\hat{\boldsymbol{l}}$ and its region.

2.1. Node Model

A network of N random-deployed wireless smart motes (referred to as "nodes") is considered, programmed for collaborative detection of a desired event occurrence. In fact, there are two conditions: the normal condition in which no event has occurred (denoted by hypothesis $\mathcal{H}_0$), and event occurrence (denoted by hypothesis $\mathcal{H}_1$). Node i, $i \in \{1, \ldots, N\}$ observes a common phenomenon of interest (POI) for detection of a desired scalar parameter $\theta(\boldsymbol{l})$ originating from the event location $\boldsymbol{l}$ according to the observation model:

$$z_i = \begin{cases} g(\theta(\boldsymbol{l})) + v_i, & \mathcal{H}_1 \\ v_i, & \mathcal{H}_0 \end{cases} \tag{1}$$

where v_i is a zero-mean additive white Gaussian noise (AWGN) with variance σ^2 (i.e., $v_i \sim \mathbb{N}(0, \sigma^2)$), and $g(.)$ is a mapping function (e.g., a function indicating the attenuation of an acoustic signal originating from position $\boldsymbol{l}$). It is assumed that the noises of nodes, v_i, are uncorrelated both temporally and spatially.

Each node i takes a local decision u_i about event occurrence based on an optimal decision rule—later discussed in Section 3.3—and sends it to an FC (Figure 1). Here, note that the scalar observation model (1) has been adopted just for simplicity in our discussions. If the desired parameter (signal) is multi-dimensional, just the decision rule would simply change. Nevertheless, we adopt the scalar observation model since detection is not the focus of our study.

2.2. Network Model

We denote the location of node $i, i \in \{1, \ldots, N\}$ by $\boldsymbol{x}_i$ and collect all locations in $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. In order to detect the event region, it is assumed that the nodes' locations are known by FC. The positions may be obtained by using an appropriate localization method such as those presented in [28,29].

As illustrated in Figure 1, a parallel configuration has been adopted in this paper. The parallel configuration is the most popular configuration in the literature. However, nodes may transmit their decisions to FC indirectly through intermediate nodes in a hop-by-hop manner (since their communication range is usually limited). This configuration can be represented by the parallel configuration as well if the communication channels are assumed to be error-free [30–32].

2.3. Communication Channel Model

The communication channels between nodes and FC are assumed to be ideal and error-free. In practice, there are two sources of error: (i) erroneous local decisions, which are due to either local false alarms or misses of nodes, and (ii) erroneous received decisions due to the faulty nature of wireless channels. We integrate both error sources into just the first type and ignore the communication error rate since it does not affect the implementation and evaluation of our proposed algorithms at all.

2.4. Problem Statement

After the event occurrence is detected by FC, the problems are to:

1. estimate the event location $\boldsymbol{l}$, and
2. divide the region into event and non-event regions.

3. Backgrounds

In this section, we review briefly the SVM (Section 3.1) and TWSVM (Section 3.2) learning algorithms. The section ends with a refresher of detection over WSN (Section 3.3), needed for the discussion of the proposed methods.

3.1. Support Vector Machine (SVM)

The support vector machines (SVMs) are a popular class of supervised learning algorithms. A comprehensive tutorial on the topic could be found in [33]. In its original formulation, SVM attempts to separate two classes of data by a hyperplane whose distance from the data of each class is maximized.

Let us denote the training set by $\boldsymbol{x} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ where the class of each sample $\boldsymbol{x}_i$ is represented by $y_i \in \{-1, 1\}$. The goal of SVM is to separate the two classes of data by a hyperplane $y = \boldsymbol{w}^T\boldsymbol{x} + b$ whose distance with the nearest samples is maximized. To that end, the following constrained optimization problem must be solved [33]:

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T\boldsymbol{x} + 1\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{2}$$

where $\{\xi_i\}$s are slack variables used for non-separable samples, and $C > 0$ is a penalty term used for allowing some limited misclassifications. More specifically, as shown in Figure 2a, the SVM algorithm attempts to separate the two classes by maximizing the margin around the optimum hyperplane. However, there would be borderline samples in most practical cases that make classification challenging (Figure 2b). The slack variables ξ_i are used for taking these borderline samples into account by imposing a penalty for them. $\xi_i = 0$ for samples out of the margin, $0 < \xi_i < 1$ for samples within the margin, and $\xi_i > 1$ for samples that are misclassified.

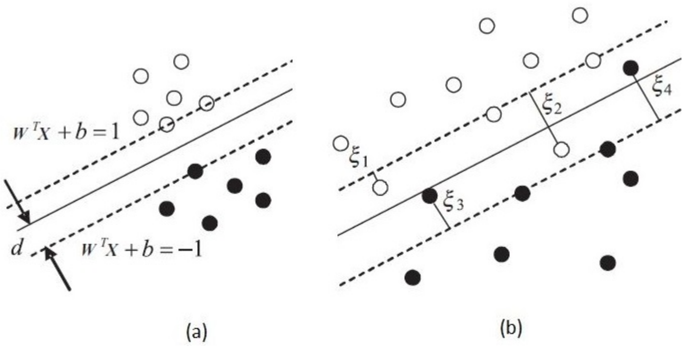

Figure 2. (**a**) Support Vector Machine (SVM) procedure for separating two classes of data by a hyperplane whose distance from the nearest samples of the classes (i.e., the margin *d*) is maximized. (**b**) SVM for the non-separable case in which the samples within the margin are penalized [33].

The optimization problem (2) can be solved by resorting to its dual problem and forming the Lagrange objective function as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{YRY\alpha} - \boldsymbol{\alpha}^T \mathbf{1} \\ \text{s.t.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \le \alpha_i \le C, \quad i = 1, \cdots, N \end{aligned} \tag{3}$$

where $\boldsymbol{Y}$ is a diagonal matrix with $\boldsymbol{Y}_{ii} = y_i$, $\boldsymbol{\alpha}$ is the column vector of the Lagrange coefficients, $\mathbf{1}$ is an $N \times 1$ vector of all elements 1, and $\boldsymbol{R}$ is the matrix of the inner products of the samples with its elements given by:

$$R_{i,j} = \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{4}$$

Now, a quadratic optimization problem has been obtained that can be solved by using appropriate functions in related libraries of programming languages, such as the function quadprog(.) in MATLAB. After solving (3) for $\boldsymbol{\alpha}$, the solution to the optimum hyperplane is given by:

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \\ b &= y_i - \boldsymbol{w}^T \boldsymbol{x}_i \end{aligned} \tag{5}$$

The final decision rule for each input vector $\boldsymbol{x}$ is now:

$$f(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b) \tag{6}$$

where sign(.) is the sign function.

Though the SVM algorithm discussed above is applicable just in cases in which the two classes can be separated by a linear hyperplane, it can be used in nonlinear cases via the Mercer kernels.

Using Mercer kernels, the data is mapped into a higher dimensional space wherein the two classes are linearly separable. To that end, one should simply replace the inner product matrix $\boldsymbol{R}$ in (3) with an appropriate kernel matrix. As an instance, a popular kernel—also used in this paper—is the Gaussian kernel. Using the Gaussian kernel, the kernel matrix is built with its elements given by:

$$K_{i,j} \triangleq K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{7}$$

Then, the decision function (6) generalizes to:

$$f(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b) \tag{8}$$

3.2. Twin Support Vector Machine (TWSVM)

In this subsection, Twin Support Vector Machine (TWSVM) [25] is discussed as a popular and accurate method of data classification. TWSVM formulation is similar to that of SVM except that in TWSVM, all the data points do not emerge in the constraints, simultaneously. Furthermore, TWSVM is proved to be faster than SVM as it solves two smaller sized QPPs [25].

Denoting the data points in class $+1$ and class -1 by matrices $\boldsymbol{A} \in R^{m_1 \times n}$ and $\boldsymbol{B} \in R^{m_2 \times n}$, respectively, ($m_1 + m_2 = N$ and n is the size of each feature) with $\boldsymbol{A}_i$ and $\boldsymbol{B}_i \in R^n$ denoting the ith row of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. The linear TWSVM, unlike SVM, finds a pair of nonparallel hyperplanes as follows:

$$\boldsymbol{w}_1^T \boldsymbol{x} + b_1 = 0 \quad \text{and} \quad \boldsymbol{w}_2^T \boldsymbol{x} + b_2 = 0 \tag{9}$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2 \in R^n$, and $b_1, b_2 \in R$. Each hyperplane of TWSVM is close to the data points of one of the two classes while it is distant from the data points of the other class. Therefore, the formulation of TWSVM can be stated as follows:

$$\begin{aligned} \min_{\boldsymbol{w}_1, b_1, \boldsymbol{q}_1} \quad & \|\boldsymbol{A}\boldsymbol{w}_1 + \mathbf{1}b_1\|^2 + c_1 \mathbf{1}^T \boldsymbol{q}_1 \\ \text{s.t.} \quad & -(\boldsymbol{B}\boldsymbol{w}_1 + \mathbf{1}b_1) + \boldsymbol{q}_1 \geq \mathbf{1} \\ & \boldsymbol{q}_1 \geq 0 \end{aligned} \tag{10}$$

$$\begin{aligned} \min_{\boldsymbol{w}_2, b_2, \boldsymbol{q}_2} \quad & \|\boldsymbol{B}\boldsymbol{w}_2 + \mathbf{1}b_2\|^2 + c_2 \mathbf{1}^T \boldsymbol{q}_2 \\ \text{s.t.} \quad & (\boldsymbol{A}\boldsymbol{w}_2 + \mathbf{1}b_2) + \boldsymbol{q}_2 \geq \mathbf{1} \\ & \boldsymbol{q}_2 \geq 0 \end{aligned} \tag{11}$$

where c_1, c_2 are positive parameters, and $\boldsymbol{q}_1, \boldsymbol{q}_2$ are the vectors of the related slack variables. It is clear that the purpose of TWSVM is to solve two QPPs (10) and (11). Each QPP in the TWSVM pair represents a classic SVM formulation, with the exception that not all data points show up in the constraints of either problem (see Figure 3).

The corresponding Wolfe dual problems can be obtained respectively as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^T \boldsymbol{\alpha} - \tfrac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{G} (\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{G}^T \boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq c_1, i \in \{1, \ldots, m_2\} \end{aligned} \tag{12}$$

$$\begin{aligned} \max_{\gamma} \quad & \mathbf{1}^T\gamma - \tfrac{1}{2}\gamma^T \boldsymbol{H}(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{H}^T\gamma \\ \text{s.t.} \quad & 0 \le \gamma_i \le c_2, i \in \{1,\ldots,m_1\} \end{aligned} \tag{13}$$

where $\boldsymbol{\alpha}$ and γ are Lagrange multipliers, $\boldsymbol{H} \triangleq [\boldsymbol{A}\ \mathbf{1}]$, and $\boldsymbol{G} \triangleq [\boldsymbol{B}\ \mathbf{1}]$.

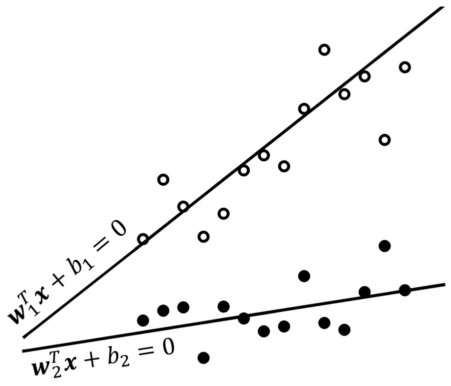
$\boldsymbol{w}_1^T\boldsymbol{x} + b_1 = 0$
$\boldsymbol{w}_2^T\boldsymbol{x} + b_2 = 0$

Figure 3. Geometric interpretation of Twin Support Vector Machine (TWSVM).

It can be shown that the solutions to (12) and (13) are given by $[w_1\ \ b_1]^T = -(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{G}^T\boldsymbol{\alpha}$ and $[w_2\ \ b_2]^T = (\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{H}^T\gamma$.

The matrices $\boldsymbol{H}^T\boldsymbol{H}$ and $\boldsymbol{G}^T\boldsymbol{G}$ are positive semidefinite, thus they may be singular. Therefore, in order to avoid ill-conditioned cases, the inverse matrices $(\boldsymbol{G}^T\boldsymbol{G})^{-1}$and $(\boldsymbol{H}^T\boldsymbol{H})^{-1}$ are approximately replaced by $(\boldsymbol{G}^T\boldsymbol{G} + \delta\boldsymbol{I})^{-1}$and $(\boldsymbol{H}^T\boldsymbol{H} + \delta\boldsymbol{I})^{-1}$ respectively, where δ is a very small positive scalar.

The nonlinear TWSVM can be expressed as follows:

$$\begin{aligned} \min_{w_1,b_1,q_1} \quad & \|\boldsymbol{K}(\boldsymbol{A},\boldsymbol{C}^T)w_1 + \mathbf{1}b_1\|^2 + c_1\mathbf{1}^T\boldsymbol{q}_1 \\ \text{s.t.} \quad & -(\boldsymbol{K}(\boldsymbol{B},\boldsymbol{C}^T)w_1 + \mathbf{1}b_1) + \boldsymbol{q}_1 \ge \mathbf{1} \\ & \boldsymbol{q}_1 \ge 0 \end{aligned} \tag{14}$$

$$\begin{aligned} \min_{w_2,b_2,q_2} \quad & \|\boldsymbol{K}(\boldsymbol{B},\boldsymbol{C}^T)w_2 + \mathbf{1}b_2\|^2 + c_2\mathbf{1}^T\boldsymbol{q}_2 \\ \text{s.t.} \quad & (\boldsymbol{K}(\boldsymbol{A},\boldsymbol{C}^T)w_2 + \mathbf{1}b_2) + \boldsymbol{q}_2 \ge \mathbf{1} \\ & \boldsymbol{q}_2 \ge 0 \end{aligned} \tag{15}$$

where $\boldsymbol{C}^T \triangleq [\boldsymbol{A}\ \ \boldsymbol{B}]^T$, and $\boldsymbol{K}(.,.)$ is the kernel matrix of an appropriately chosen type (the Gaussian kernel in (7) here for its elements). The Wolfe dual problems of (14) and (15) to obtain the following hypersurfaces

$$\boldsymbol{K}(\boldsymbol{x}^T,\boldsymbol{C}^T)w_1 + b_1 = 0 \ \text{ and } \ \boldsymbol{K}(\boldsymbol{x}^T,\boldsymbol{C}^T)w_2 + b_2 = 0 \tag{16}$$

are respectively as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathbf{1}^T\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{G}'(\boldsymbol{H}'^T\boldsymbol{H}')^{-1}\boldsymbol{G}'^T\boldsymbol{\alpha} \\ \mathring{s}.t. \quad & 0 \le \alpha_i \le c_1, i \in \{1,\ldots,m_2\} \end{aligned} \tag{17}$$

$$\begin{aligned} \max_{\gamma} \quad & \mathbf{1}^T\gamma - \tfrac{1}{2}\gamma^T\boldsymbol{H}'(\boldsymbol{G}'^T\boldsymbol{G}')^{-1}\boldsymbol{H}'^T\gamma \\ \text{s.t.} \quad & 0 \le \gamma_i \le c_2, i \in \{1,\ldots,m_1\} \end{aligned} \tag{18}$$

where $\boldsymbol{H}' \triangleq [\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{C}^T) \quad \mathbf{1}]$ and $\boldsymbol{G}' \triangleq [\boldsymbol{K}(\boldsymbol{B}, \boldsymbol{C}^T) \quad \mathbf{1}]$. Solving the above problems gives $[\boldsymbol{w}_1 \;\; b_1]^T = -(\boldsymbol{H}'^T\boldsymbol{H}' + \delta\boldsymbol{I})^{-1}\boldsymbol{G}'^T\boldsymbol{\alpha}$, and $[\boldsymbol{w}_2 \;\; b_2]^T = (\boldsymbol{G}'^T\boldsymbol{G}' + \delta\boldsymbol{I})^{-1}\boldsymbol{H}'^T\boldsymbol{\gamma}$. Then, the distance of each input data to each of the two obtained hypersurfaces determines its class. In other words, for each input data $\boldsymbol{x}$, its class y is determined by:

$$y = 2\left(\arg\min_{j\in\{1,2\}} \left|\boldsymbol{w}_j^T\boldsymbol{x} + b_j\right| - 1.5\right), i \in \{1, \ldots, N\} \tag{19}$$

3.3. Detection over Sensor Networks

In this subsection, the theory of detection and its implementation over sensor networks are reviewed. A comprehensive tutorial on the topic is [20].

Detection is a basic task of WSNs in many applications [34,35] where a final decision about either an event occurrence (labeled as "hypothesis $\mathcal{H}_1$") or no event occurrence (denoted by "hypothesis $\mathcal{H}_0$") must be taken. This final decision is taken by FC via fusing the data of the network nodes.

The network nodes may send either raw or processed observations to FC. While sending raw measurements imposes a considerable communication burden on the network, the nodes are usually programmed to make a local decision and inform FC about their decision by sending just one bit. To have this practically popular scheme—known as "distributed detection"—implemented, two types of decision rules must be designed: a local decision rule for each node, and a decision fusion rule for FC. These two kinds of decision rules are discussed in what follows.

3.3.1. Local Decision Rule

Nodes of the network observe a desired signal θ according to model (1). In practical scenarios, there is no information about the specifications of the source signal (e.g., the location and the strength of the acoustic signal are not known). In these cases, it is proved that the optimal decision rule for each node would be the likelihood ratio test (LRT) if the statistical information of the sensing noise (v_i in (1)) is available [36]. Accordingly, the optimal decision rule for node i with observation model (1) is given by [36]:

$$u_i = \begin{cases} 0, & z_i^2 < \tau_i \\ 1, & z_i^2 > \tau_i \end{cases} \tag{20}$$

with $u_i = 0$ ($u_i = 1$) indicating $\mathcal{H}_0$ ($\mathcal{H}_1$), and τ_i being the detection threshold that is given based on a local false alarm rate (i.e., $\Pr(u_i = 1|\mathcal{H}_0)$).

3.3.2. Fusion Rule

The network nodes send their decision to FC where a decision fusion rule should be adopted in order to have a final decision. One simple and popular one is the counting rule [37] in which the sum of received decisions is simply compared against a threshold:

$$\Lambda \triangleq \sum_{i=1}^{N} u_i \underset{H_0}{\overset{H_1}{\gtrless}} t \tag{21}$$

where N indicates the network size, and t is the detection threshold which is obtained according to a desired network false alarm rate. Under a common threshold for sensors and the same noise statistics (i.e., in homogeneous WSNs), Λ under $\mathcal{H}_0$ (i.e., $\Lambda|\mathcal{H}_0$) is binomial distributed while it follows a Poisson-binomial distribution in more general cases [38]. The threshold t is computed according to this distribution. While simple, the counting rule maintains robustness [39,40] and can reach almost the optimum detection performance in large network sizes [20] (i.e., in sufficiently large values of N). There are modifications of the counting rule such as LVDF [41] and WDF [35,42]. Each node in LVDF modifies its decision based on the majority of the decisions of its neighbors. In WDF, the decision of each node is weighted based on the node's sensed signal-to-noise ratio (SNR) and then the weighted decisions are counted. Moreover, the network's detection performance can be improved by considering the correlation among the nodes' decisions [43,44].

4. Learning Methods for Source Localization

In this section, we propose two learning-based methods for both source localization and obtaining the nodes affected by the event. The complexity of the methods and their designing methodology are discussed as well.

4.1. Region of Event Detection by SVM (Red-S)

In this section, our first proposed method of source localization in WSNs is presented. Having it implemented in a WSN, we show that the region of event can be recognized. This method is referred to as "*Red-S*" which is carried out by the following steps:

1. The network nodes observe ROI in order to detect a desired event (e.g., a desired target) when it occurs by using the decision rule (20). Then, they send their decisions to an FC where the final decision is taken by exploiting an appropriate fusion rule such as the counting rule (21). It is assumed that the network size is sufficiently large so that the overall detection performance of the network is optimum (i.e., there is neither a false alarm nor a miss).
2. Upon event detection, FC runs the SVM algorithm in order to detect the region of the event (i.e., the nodes in vicinity of the event) as follows:
    (a) The locations of the nodes are considered as the training set $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with $\boldsymbol{x}_i$ being the location of node i.
    (b) The decision of each node denotes its class according to the following mapping rule:
    $$\begin{aligned} u_i = 0 &\rightarrow y_i = -1 \\ u_i = 1 &\rightarrow y_i = 1 \end{aligned} \tag{22}$$
    where $i \in \{1, \ldots, N\}$.
    (c) The Gaussian kernel (7) is adopted for constituting the kernel matrix $\boldsymbol{K}$ that replaces the matrix $\boldsymbol{R}$ in the optimization problem (3). Designing parameter σ of the Gaussian kernel is elaborated in Section 4.3.1.
    (d) Having SVM solved, the coefficient vector $\boldsymbol{\alpha}$ is obtained based on which the classifier parameters $\boldsymbol{w}$ and b are obtained.
    (e) The event region is obtained by applying the nodes' locations to the obtained classifier as follows:
    $$\boldsymbol{y} = \text{sign}\left(\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} + b\boldsymbol{1}\right) \tag{23}$$
    where $y_i = 1$ denotes that node i is in the even region while it is not if $y_i = -1$. Note that the above relation is the vectorized version of (8).
3. The location of the event is estimated by averaging the locations of the nodes in the region of the event. In other words, the centroid of the nodes in the vicinity of the event is considered as the location of the event. More specifically, denoting the set of the nodes in event region by $\mathcal{E} \triangleq \{\boldsymbol{x}_i : y_i = 1, i = 1, \ldots, N\}$, the estimation of the event location is given by:
$$\hat{\boldsymbol{l}} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} \boldsymbol{x}_i \tag{24}$$
where $|\mathcal{E}|$ denotes the cardinality of $\mathcal{E}$.

In summary, the locations of nodes are used as the training set together with the nodes' decisions as their classes. After having the network trained, the nodes' locations are applied to the obtained classifier. The result gives the region of event with its centroid as the estimation of the event's location.

4.2. Region of Event Detection by TWSVM (Red-T)

Red-S gives both event location and its region after the event occurrence is detected by FC. To improve the accuracy of classification of ROI into event and non-event regions, the more advanced TWSVM algorithm is applied. However, TWSVM is more sensitive to design parameters—as discussed later—and needs a priori estimation of event location in order to yield accurate outcomes. This estimation can be provided by Red-S. In other words, the classification performance of Red-S is improved by applying TWSVM on the network. The overall algorithm is referred to as "Red-T" and performs the following steps after Red-S result is out:

1. The parameters of TWSVM are computed based on the classification result of Red-S as will be elaborated in Section 4.3.2.
2. The TWSVM classification algorithm with its parameters computed in the previous step is applied on the nodes' locations as the training set. The class of each node is determined according to the mapping rule given in (22).
3. TWSVM gives two hypersurfaces based on the training set by solving (17) and (18).
4. The nodes are assigned to regions of event and non-event depending on to which of the two hypersurfaces given by (17) and (18) are closer. In other words, the class of node i is updated according to:

$$y_i = 2\left(\arg\min_{j\in\{1,2\}} \left|\boldsymbol{w}_j^T\boldsymbol{x}_i + b_j\right| - 1.5\right), i \in \{1,\ldots,N\} \tag{25}$$

   where $y_i = 1$ denotes that node i is in the region of event while it is not in the region of event if $y_i = -1$.
5. Similarly as in Res-S, the location of event is obtained by averaging all nodes in event region, i.e., by (24).

As an instance of how Red-S and Red-T work, see Figure 4 where the classification results of applying Red-S and Red-T to the network condition in Figure 4a has been illustrated in Figure 4b,c, respectively. As shown, both event location and its region have been detected in both methods; though the performance has been improved by applying Red-T. Note that the nodes with false alarms are ignored by Red-S and Red-T while the decisions of the nodes which are in the vicinity of the event but miss the detection are corrected.

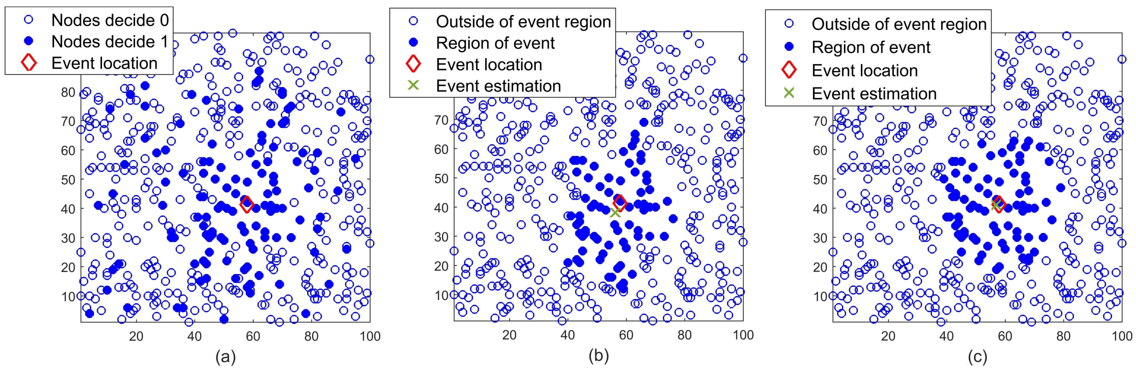

Figure 4. (**a**) Distributed detection in a wireless sensor network with 500 nodes randomly deployed in a 100 m × 100 m region and local false alarm probability $p_{fa} = 0.1$. (**b**) Event localization after applying Red-S. (**c**) Event localization after applying Red-T.

4.3. Designing Parameters

4.3.1. Red-S Parameters

During implementation of Red-S, the value of σ in the Gaussian kernel (7) is of importance. Its value should be chosen so that a reasonable kernel value is obtained. Therefore, it can be considered as a function of the average distance between nodes, or simply a function of network density. For example, the average distance between nodes is less in dense networks, so σ should be adjusted to be less as well.

4.3.2. Red-T Parameters

The performance of TWSVM depends on the choices of the kernel function parameters. The optimal values for the parameters are determined by the grid search method with the aim of minimizing the distance between the estimated location of event by TWSVM and that of Red-S.

To alleviate the computational burden, the settings $c_1 = c_3$ and $c_2 = c_4$ in (17) and (18) are used. As in [45], the grid values for c_1, c_2, c_3, and c_4 are considered to be in $\{10^i | i = -10, -4, ..., 10\}$ and the kernel width σ chosen from the range $\{10^i | i = -5, -4, ..., 1\}$ for each dataset.

4.4. Edge Effect Correction

As discussed in previous sections, the centroid of the nodes in the event region, i.e., (24), is considered as the source location. Since the centroid is never located in the edges of the region under surveillance (ROI), Red-S and Red-T do not perform well at the edges of ROI; we refer to this fault as the *edge effect*.

The edge effect is alleviated by adding (or subtracting) a random number to the already obtained estimation of the source location. More specifically, if a portion of the nodes in event region, say 20 percent of them, are among the nodes with the lowest 2nd coordination (e.g., the nodes of the left edge of the network shown in Figure 1), the event may have been occurred at the edge. Thus the first coordination of $\hat{\boldsymbol{l}}$ should be corrected by:

$$\hat{l}(1) := \hat{l}(1) - 2u \tag{26}$$

where $:=$ indicates modification (like assignment operator in programming languages), and u is a uniformly distributed random variable between 0 and $\hat{l}(1)$, i.e., $u \sim \mathcal{U}(0, \hat{l}(1))$. The same correction mechanism may be applied for other edges. Such correction is validated in Section 5.

4.5. Computational Complexity

SVM requires to solve a QPP with inequality constraints. It is well-known that for a training data set of size N, the computational complexity of SVM is $\mathcal{O}\left(N^3\right)$ [46,47]. On the other hand, as discussed in Section 3.2, TWSVM solves a pair of QPPs instead of a single complex QPP of SVM. If the number of patterns in each class approximately equals to $N/2$, the complexity of TWSVM is $\mathcal{O}\left(2(N/2)^3\right)$ [25]. In other words, TWSVM is four times faster than SVM. Accordingly, the complexities of Red-S (involving SVM) and Red-T (involving both SVM and TWSVM) are $\mathcal{O}\left(N^3\right)$ and $\mathcal{O}\left(\frac{5}{4}N^3\right)$, respectively.

Note that both outstanding source localization methods based on statistical signal processing, MI-based and PCRLB-based methods, are iterative. Moreover, the nodes quantize their observations into L bits and A nodes are selected in each iteration [18]. The complexity of MI-based method in *each iteration* is $\mathcal{O}\left(NL^A + ALN\right)$ [18] which grows exponentially with the network size (more precisely, with the number of nodes selected in each iteration). The PCRLB is much simpler with its complexity being $\mathcal{O}\left(ALN\right)$ [18] in each iteration. Therefore, Red-S and Red-T, while consuming much less bandwidth with sending just one bit by each node, provide complexity reduction, especially compared to the MI-based method.

5. Evaluation and Discussion

In this section, the performance of the Red-S and Red-T schemes is evaluated. To that end, a randomly deployed homogeneous WSN in a 100 m $\times$ 100 m region is considered where nodes' observations are contaminated by a standard normal AWGN. It is assumed that the network goal is to detect a target with an acoustic signal with the following isotropic model:

$$P_i = \frac{P_0}{1 + d_{ti}^a} \tag{27}$$

where P_i denotes the strength of the acoustic source at the location of node i, P_0 is the strength of the source in 1 m distance from the target, d_{ti} is the distance between node i and the target, and a is an attenuation coefficient ($a = 2$ is considered throughout our simulations). Therefore, the observation of node i can be modeled as:

$$z_i = \sqrt{P_i} + v \tag{28}$$

In simulations, $C = 0.5$ has been used as the trade-off parameter of the SVM optimization problem, and $\sigma = \frac{20000}{N}$ as the kernel parameter (i.e., $\sigma = 20$ is used for $N = 1000$). The accuracy of Red-S and Red-T for source localization has been evaluated in terms of mean squared error (MSE) defined by:

$$MSE = \mathbb{E}\left(\left\| \boldsymbol{l} - \hat{\boldsymbol{l}} \right\|^2 \right) \tag{29}$$

in which $\boldsymbol{l}$ and $\hat{\boldsymbol{l}}$ are respectively the exact and the estimated location of the target, and $\mathbb{E}(.)$ denotes the statistical expectation.

The proposed methods are compared against the fault recognition (FR) algorithm [27]. In FR, the potentially faulty decision (i.e., either false alarm or miss of the event) of each node is meant to be corrected by comparing it with the decisions of the node's neighbors. In fact, FR exploits the spacial correlation among nodes' decisions and works as follows:

1. Determine the neighbors of each node (i.e., the nodes in the communication range of each node).
2. Nodes take an initial decision based on their observations.
3. The final decision of each node is equal to the decision of majority of its neighbors.

The MSEs of the methods have been obtained via more than 5000 Monte-Carlo runs in different scenarios whose results have been depicted in Figures 5–7. In addition to FR, we also examined the naive source localization where simply the centroid of all nodes with $u_i = 1$ is considered as the estimation of the source location. However, the MSEs of this method were more than three times those of FR (more than 120 m^2 in all cases). Therefore, its performance is included just in Figure 5 where the MSEs of the methods are shown in different network sizes. Comparison of the performance of naive source localization in different network densities (Figure 5a) with that of the other three methods (Figure 5b) reveals how much the performance is improved by processing the raw decisions.

Figure 5b shows that the estimation error decreases in more dense networks. However, the decrease rate becomes negligible after a specified density. In other words, it is not advantageous to employ more network nodes after a specified density. For example, in our scenario in Figure 5b, the difference between the MSE of a 600-node and a 1000-node networks is less than 4 m^2. Figure 5b also shows that MSE of Red-S yields a more accurate estimation of the event location in a less dense network. This is because the parameters of Red-T are more sensitive to the dynamics of networks. In addition, note that the grid search of the TWSVM algorithm has been confined in order to speed up the running time. In fact, Red-S is more robust than Red-T because there is just one parameter to be adjusted.

It has been shown in Figure 6 that the estimation error is lower in more values of SNR. In addition, Figure 7 shows how local false alarm rates affect the overall target tracking error. Note that nodes send more data (and hence consume more energy) in more false alarm rates. Therefore, the false alarm rate must be as low as possible. However, the very low local false alarm rate decreases the detection probability of the network as well. Therefore, there is a trade-off here.

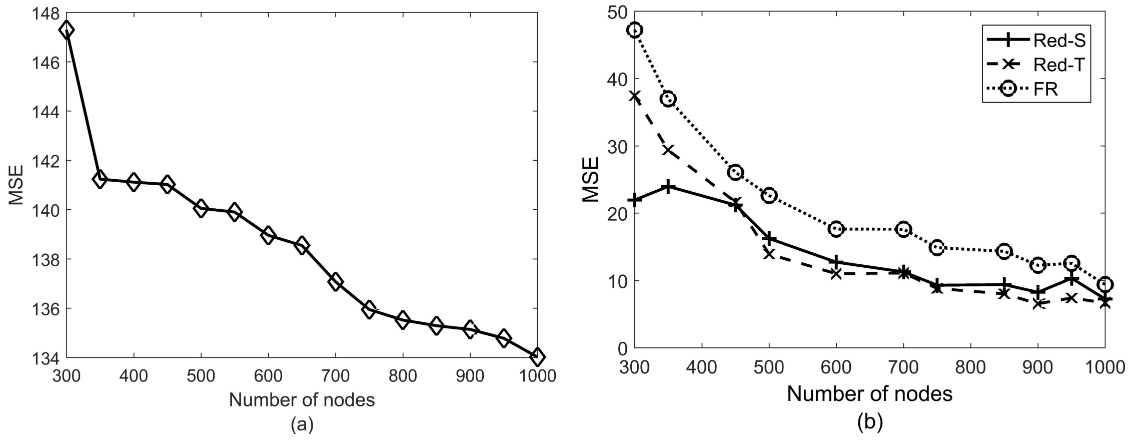

Figure 5. (**a**) Mean squared error (MSE) of naive source localization. (**b**) MSE of Red-S, Red-T, and FR in different number of nodes (N). $P_0 = 1000$ and the local false alarm probability $p_{fa} = 0.1$ have been considered in simulations.

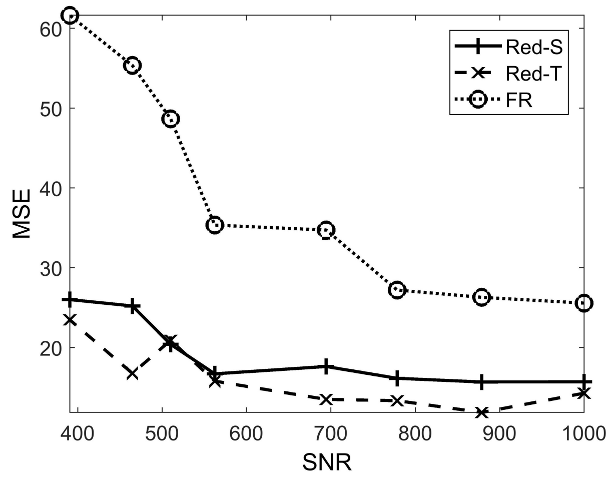

Figure 6. Mean squared error (MSE) of Red-S, Red-T, and FR in different values of signal-to-noise ratio (SNR) in a 500-node network. The local false alarm probability $p_{fa} = 0.1$ has been considered in simulations.

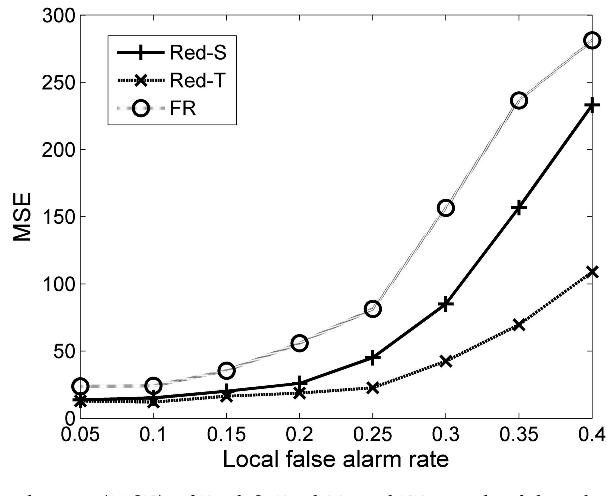

Figure 7. Mean squared error (MSE) of Red-S, Red-T, and FR vs. the false alarm probability of local nodes in a 500-node network. $P_0 = 1000$ has been considered in simulations.

In another scenario, shown in Figure 8, the accuracy of Red-S in tracking a moving target has been assessed. In this scenario, a target moves from the lowest left corner to the highest right corner in a constant speed. Figure 8 shows the average of 10 tracking performances. As is shown, the tracking performance is better in more dense networks. In addition, note that the accuracy is acceptable in even networks with low densities. Another point is that the performance is not appropriate in corners since the centroid of the region of the event is considered as the location of the event in Red-S which is never located at the edges of ROI. However, as shown in Figure 8b, the edge effect is alleviated appropriately, especially in denser cases, by applying the correction method discussed in Section 4.4.

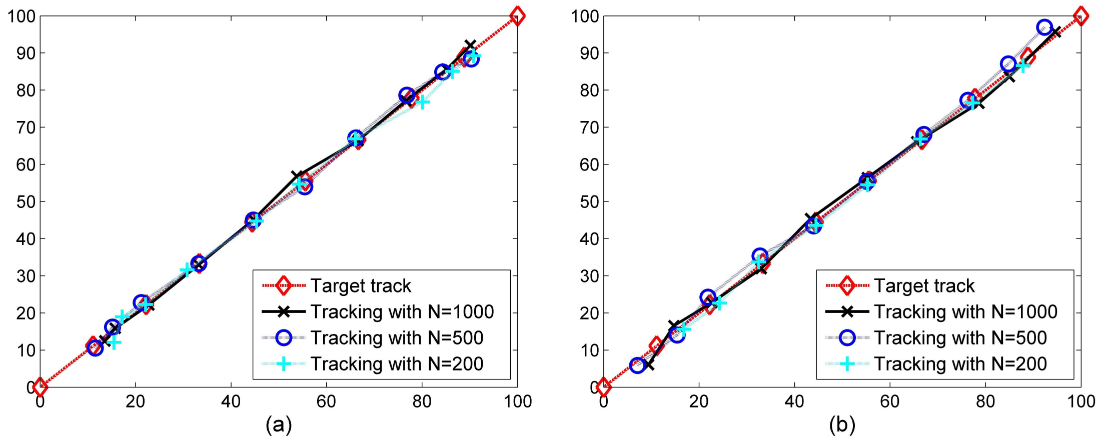

Figure 8. Accuracy of Red-S in tracking a moving target through a network in a 100 m $\times$ 100 m region. N indicates the number of network nodes. $P_0 = 1000$ and local false alarm rate p_{fa} have been considered in the simulations. (**a**) Without edge effect correction. (**b**) With edge effect correction.

6. Conclusions and Future Directions

In this paper, two methods of source localization were proposed for implementing in a WSN with low-cost nodes. In our proposed methods, the network firstly detects the region of the desired event and then computes the event location by averaging the locations of all nodes in the vicinity of the event. To reach that, a combination of distributed detection and learning algorithms is exploited. In fact, the source localization is carried out in three phases: (i) Network nodes send their decisions about event occurrence to FC. (ii) Upon event detection, the FC classifies the nodes into two classes of nodes in "event region" and nodes in "non-event region". (iii) The average of the locations of the nodes in the region of the event is considered as the event location.

The region of the event detection is specified by using SVM and TWSVM learning methods. Therefore, the proposed methods were referred to as "Red-S" and "Red-T", respectively. By simulating different scenarios, it was shown that Red-S and Red-T work appropriately in target localization, especially in dense networks. However, the proposed methods are capable of localizing just one event. Applying learning methods, such as the k-mean clustering method, for multi-source localization may be considered as a future work.

Finally, note that though the proposed methods were applied on the WSN framework, they are applicable to every framework with false alarms and misses. One such example is medical health tests. The methods can also be applied on outlier detection applications.

Author Contributions: Conceptualization, S.H.J.; methodology, S.H.J.; software, S.H.J. and H.M.; investigation, S.H.J. and H.M.; resources, S.H.J.; writing—original draft preparation, S.H.J.; writing—review and editing, D.C.; supervision, D.C.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

WSN	Wireless Sensor Network
IoT	Internet of Things
ML	Machine Learning
QPP	Quadratic Programming Problem
SVM	Support Vector Machine
TWSVM	Twin SVM
FC	Fusion Center
AWGN	Additive White Gaussian Noise
SNR	Signal-to-noise ratio
FR	Faulty Recognition algorithm
Red-S	Region event detection using SVM
Red-T	Region event detection using TWSVM
ROI	Region of Interest

References

1. Muzet, A. Environmental noise, sleep and health. *Sleep Med. Rev.* **2007**, *11*, 135–142. [CrossRef]
2. de Kluizenaar, Y.; Janssen, S.A.; van Lenthe, F.J.; Miedema, H.M.; Mackenbach, J.P. Long-term road traffic noise exposure is associated with an increase in morning tiredness. *J. Acoust. Soc. Am.* **2009**, *11*, 135–142. [CrossRef]
3. Miedema, H.M.; Oudshoorn, C.G. Annoyance from transportation noise: Relationships with exposure metrics DNL and DENL and their confidence intervals. *Environ. Health Perspect.* **2009**, *109*, 409–416. [CrossRef] [PubMed]
4. Babisch, W.; Beule, B.; Schust, M.; Kersten, N.; Ising, H. Traffic noise and risk of myocardial infarction. *Epidemiology* **2005**, *16*, 33–40. [CrossRef] [PubMed]
5. Lercher, P.; Evans, G.W.; Meis, M. Ambient noise and cognitive processes among primary schoolchildren. *Environ. Behav.* **2003**, *35*, 725–735. [CrossRef]
6. Chetoni, M.; Ascari, E.; Bianco, F.; Fredianelli, L.; Licitra, G.; Cori, L. Global noise score indicator for classroom evaluation of acoustic performances in LIFE GIOCONDA project. *Noise Mapp.* **2016**, *3*. [CrossRef]
7. Van Kempen, E.; Babisch, W. The quantitative relationship between road traffic noise and hypertension: A meta-analysis. *J. Hypertens.* **2012**, *30*, 1075–1086. [CrossRef]
8. Zambon, G.; Roman, H.; Smiraglia, M.; Benocci, R. Monitoring and prediction of traffic noise in large urban areas. *Appl. Sci.* **2018**, *8*, 251. [CrossRef]
9. Licitra, G.; Ascari, E.; Fredianelli, L. Prioritizing Process in Action Plans: A Review of Approaches. *Curr. Pollut. Rep.* **2017**, *3*, 151–161. [CrossRef]
10. Ciuonzo, D.; Salvo Rossi, P. Distributed detection of a non-cooperative target via generalized locally-optimum approaches. *Inf. Fusion* **2017**, *36*, 261–274. [CrossRef]
11. Ciuonzo, D.; Salvo Rossi, P.; Willett, P. Generalized Rao Test for Decentralized Detection of an Uncooperative Target. *IEEE Signal Process. Lett.* **2017**, *24*, 678–682. [CrossRef]
12. Manyika, J.; Durrant-Whyte, H. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*; Ellis Horwood: New York, NY, USA, 1994.
13. Julier, S.J. Fusion without independence. In Proceedings of the IET Seminar on Target Tracking and Data Fusion: Algorithms and Applications, Birmingham, UK, 15–16 April 2008 .
14. Chong, C.Y.; Mori, S.; Chang, K. Distributed Multitarget Multisensor Tracking. In *Multitarget-Multisensor Tracking: Advanced Applications*; Bar-Shalom, Y., Ed.; Artech House: Norwood, MA, USA, 1990; Chapter 8.
15. Chang, K.C.; Saha, R.K.; Bar-Shalom, Y. On optimal track-to-track fusion. *IEEE Trans. Aerosp. Electron. Syst.* **1997**, *33*, 1271–1276. [CrossRef]
16. Battistelli, G.; Chisci, L.; Farina, A.; Graziano, A. Consensus CPHD Filter for Distributed Multitarget Tracking. *IEEE J. Sel. Top. Signal Process.* **2013**, *7*, 508–520. [CrossRef]
17. Williams, J.L.; Fisher, J.W.; Willsky, A.S. Approximate Dynamic Programming for Communication-Constrained Sensor Network Management. *IEEE Trans. Signal Process.* **2007**, *55*, 4300–4311. [CrossRef]

18. Masazade, E.; Niu, R.; Varshney, P.K.; Keskinoz, M. Energy Aware Iterative Source Localization for Wireless Sensor Networks. *IEEE Trans. Signal Process.* **2010**, *58*, 4824–4835. [CrossRef]
19. Zuo, L.; Niu, R.; Varshney, P.K. Conditional Posterior Cramer Rao Lower Bounds for Nonlinear Sequential Bayesian Estimation. *IEEE Trans. Signal Process.* **2011**, *59*, 1–14. [CrossRef]
20. Javadi, S.H. Detection over sensor networks: A tutorial. *IEEE Aerosp. Elect. Syst. Mag.* **2016**, *31*, 2–18. [CrossRef]
21. Alsheikh, M.A.; Lin, S.; Niyato, D.; Tan, H. Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1996–2018. [CrossRef]
22. Vapnik, V.N. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998.
23. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
24. Ketabchi, S.; Moosaei, H.; Razzaghi, M.; Pardalos, P.M. An improvement on parametric ν-support vector algorithm for classification. *Ann. Oper. Res.* **2017**, 1–14. [CrossRef]
25. Jayadeva; Khemchandani, R.; Chandra, S. Twin Support Vector Machines for Pattern Classification. *IEEE Trans. Pattern Anal. Mach. Intel.* **2007**, *29*, 905–910. [CrossRef]
26. Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*; Cambridge University Press: Cambridge, UK, 2000.
27. Krishnamachari, B.; Iyengar, S. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Trans. Comput.* **2004**, *53*, 241–250. [CrossRef]
28. Liu, C.; Fang, D.; Yang, Z.; Jiang, H.; Chen, X.; Wang, W.; Xing, T.; Cai, L. RSS Distribution-Based Passive Localization and Its Application in Sensor Networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 2883–2895. [CrossRef]
29. Gustafsson, F.; Gunnarsson, F.; Lindgren, D. Sensor models and localization algorithms for sensor networks based on received signal strength. *EURASIP J. Wirel. Commun. Netw.* **2012**, *2012*, 1–13. [CrossRef]
30. Viswanathan, R.; Thomopoulos, S.C.A.; Tumuluri, R. Optimal serial distributed decision fusion. *IEEE Trans. Aerosp. Elect. Syst.* **1988**, *24*, 366–376. [CrossRef]
31. Viswanathan, R.; Varshney, P.K. Distributed Detection with Multiple Sensors: Part I-Fundamentals. *Proc. IEEE* **1997**, *85*, 54–63. [CrossRef]
32. Javadi, S.H.; Peiravi, A. Reliable distributed detection in multi-hop clustered wireless sensor networks. *IET Signal Process.* **2012**, *6*, 743–750. [CrossRef]
33. Martinez-Ramon, M.; Christodoulou, C. *Support Vector Machines for Antenna Array Processing and Electromagnetics*; Morgan and Claypool: Williston, VT, USA, 2006.
34. Javadi, S.H.; Mohammadi, A. Fire detection by fusing correlated measurements. *J. Ambient Intell. Hum. Comput.* **2017**. [CrossRef]
35. Javadi, S.H.; Peiravi, A. Weighted decision fusion vs. counting rule over wireless sensor networks: A realistic comparison. In Proceedings of the 2013 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 14–16 May 2013; pp. 1–6.
36. Kay, S.M. *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*; Prentice Hall: Upper Saddle River, NJ, USA, 1998.
37. Niu, R.; Varshney, P.K. Distributed Detection and Fusion in a Large Wireless Sensor Network of Random Size. *EURASIP J. Wirel. Commun. Netw.* **2005**, *2005*, 462–472. [CrossRef]
38. Ciuonzo, D.; Romano, G.; Salvo Rossi, P. Optimality of received energy in decision fusion over Rayleigh fading diversity MAC with non-identical sensors. *IEEE Trans. Signal Process.* **2013**, *61*, 22–27. [CrossRef]
39. Ciuonzo, D.; Salvo Rossi, P. Decision Fusion With Unknown Sensor Detection Probability. *IEEE Signal Process. Lett.* **2014**, *21*, 208–212. [CrossRef]
40. Ciuonzo, D.; De Maio, A.; Salvo Rossi, P. A Systematic Framework for Composite Hypothesis Testing of Independent Bernoulli Trials. *IEEE Signal Process. Lett.* **2015**, *22*, 1249–1253. [CrossRef]
41. Katenka, N.; Levina, E.; Michailidis, G. Local Vote Decision Fusion for Target Detection in Wireless Sensor Networks. *IEEE Trans. Signal Process.* **2008**, *56*, 329–338. [CrossRef]
42. Javadi, S.H.; Peiravi, A. Fusion of weighted decisions in wireless sensor networks. *IET Wirel. Sens. Syst.* **2015**, *5*, 97–105. [CrossRef]
43. Javadi, S.H.; Mohammadi, A.; Farina, A. Hierarchical copula-based distributed detection. *Signal Process.* **2019**, *158*, 100–106. [CrossRef]

44. Javadi, S.H.; Mohammadi, A. Plackett fusion of correlated decisions. *AEU—Int. J. Electron. Commun.* **2019**, *99*, 341–346. [CrossRef]
45. Hsu, C.W.; Chang, C.C.; Lin, C.J. A practical guide to support vector classification, 2010. Available online: https://www.researchgate.net/profile/Chenghai_Yang/publication/272039161_Evaluating_unsupervised_and_supervised_image_classification_methods_for_mapping_cotton_root_rot/links/55f2c57408ae0960a3897985/Evaluating-unsupervised-and-supervised-image-classification-methods-for-mapping-cotton-root-rot.pdf (accessed on 15 April 2010).
46. Bottou, L.; Lin, C.J. Support Vector Machine Solvers. In *Large Scale Kernel Machines*; Bottou, L., Chapelle, O., DeCoste, D., Weston, J., Eds.; MIT Press: Cambridge, MA, USA, 2007.
47. Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines*; Cambridge University: Cambridge, UK, 2000.

Sample Availability: The codes of simulations are available from the authors via emails h.javadi@ub.ac.ir and s.h.javadi@ieee.org.