Neurocomputing 223 (2017) 63–76

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Hop-by-Hop Congestion Avoidance in wireless sensor networks based on genetic support vector machine

Majid Gholipour[a], Abolfazl Toroghi Haghighat[b,*], Mohammad Reza Meybodi[c]

[a] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Computer Engineering and Information Technology Department, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[c] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Communicated by Xiao-Zhi Gao

Keywords:
Wireless Sensor Networks
Congestion control
Transmission rate
Multi-classification
Support Vector Machine
Genetic Algorithm
Tukey test

ABSTRACT

Congestion in wireless sensor networks causes packet loss, throughput reduction and low energy efficiency. To address this challenge, a transmission rate control method is presented in this article. The strategy calculates buffer occupancy ratio and estimates the congestion degree of the downstream node. Then, it sends this information to the current node. The current node adjusts the transmission rate to tackle the problem of congestion, improving the network throughput by using multi-classification obtained via Support Vector Machines (SVMs). SVM parameters are tuned, using genetic algorithm. Simulations showed that in most cases, the results of the SVM network match the actual data in training and testing phases. Also, simulation results demonstrated that the proposed method not only decreases energy consumption, packet loss and end to end delay in networks, but it also significantly improves throughput and network lifetime under different traffic conditions, especially in heavy traffic areas.

1. Introduction

Wireless sensor networks (WSNs) comprise a large number of small and cheap sensor nodes with the restricted data processing and communication capabilities to sense the environment. In event-driven sensor networks, nodes operate under idle or load states. These networks are extensively used in several areas such as military surveillance, habitat monitoring, healthcare, and forest fire detection, to name a few [1,2]. The following are the main areas in which WSNs are applied: sensing the environment, collecting data, and processing them through sensor nodes cooperation and finally transferring the packets to sink nodes.

Congestion occurs when the rates of incoming and outgoing packets in the nodes are not equal [3]. When the network congestion occurs, some packets may be lost due to the limited size of the nodes buffer. This may result in the reduction of the network throughput as well as energy wasting. Therefore, congestion control is a crucial challenge in WSNs. There are several congestion control methods that use transmission rate adjustment [4].

The first study of this kind called Congestion Detection and Avoidance (CODA) was proposed by Wan et al. [5]. CODA consists of three main strategies: Congestion detection based on the received message, hop-by-hop open-loop feedback to the upstream nodes, and closed-loop adjustment of the transmission rate in the source nodes. CODA ensures that the network throughput will be appropriate through adjusting the close-loop rate. In this method, continuous monitoring of the channel causes abundant energy consumption in the involved nodes.

ECODA (Enhanced congestion detection and avoidance) [6] is a good mechanism to detect and avoid the congestion. ECODA uses three strategies in the face of congestion: At first, dual buffer thresholds are used for congestion detection. Then, the Flexible Queue Scheduler is determined for packet scheduling and finally a bottleneck-node-based source sending rate control scheme is presented to prevent the congestion. The ECODA mechanism sets q_{max}(maximum queue) to inform neighbor nodes about the occurrence of congestion, asking them to slow down the data sending rate. Two sub-queues in each node are considered, with one of them being used for a locally generated packet of a node and another one for route traffic. On the route traffic queue, packets are grouped by sources. For every source, packets are sorted by their dynamic priority from high to low. When congestion happens, the incoming packets get dropped from the tail. Therefore, the lower priority packets get dropped at first. Thus, using this scheme, a node drops some low priority packets from either of the queues when it receives high priority packets from a neighbor. In a bottleneck-node-based source sending rate control scheme, each node adjusts its data

* Corresponding author.
E-mail addresses: gholipour@qiau.ac.ir (M. Gholipour), haghighat@qiau.ac.ir (A.T. Haghighat), mmeybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.neucom.2016.10.035
Received 20 January 2016; Received in revised form 3 October 2016; Accepted 25 October 2016
Available online 28 October 2016
0925-2312/© 2016 Elsevier B.V. All rights reserved.

sending rate depending upon the neighbor's congestion information. This method is able to reduce the packet loss and improves other quality of service parameters.

DRR (Differed Reporting Rate) [7] algorithm uses different reporting rates to control congestion. This method mainly runs on each source node. It uses buffer occupancy ratio as congestion metric. If the buffer occupancy ratio is greater than the threshold value, then the congestion notification bit of outgoing packets is set and sending rate of nodes should be decreased. Otherwise, the congestion notification bit gets zero and sending rate of nodes should be increased. In this algorithm, different flow rate components are defined to adjust sensor nodes reporting rate. In the different reporting rate adjustments, sensor nodes which are near the sink, have minimum reporting rates and this rate will rise for the nodes that are away from the sink. The output rate of a node is set with newly calculated rate until next congestion notification will be sensed. The main disadvantage of this approach is that each node determines the transmission rate based on the congestion notification of neighbor and relative congestion between two neighboring nodes has no effect on the increase or decrease in transmission rate.

In [8] an algorithm was presented in order to detect congestion, using artificial neural network. In this algorithm, buffer occupancy and traffic rate were taken as the input parameters into a neural network, with the level of congestion being taken as the network output. This technique only estimates the congestion level and offers no method for congestion control. Moreover, this technique only estimates the congestion level at the sink. As a result, it is not able to estimate the congestion level at the source and relay nodes. Table 1 compares the proposed method to other congestion- related works. This table displays the properties of various congestion control protocols in WSNs. As this table shows, the data classification is the main advantage that distinguishes the proposed method from other methods. In fact, having analyzed the various classification methods, the proposed method uses support vector machine (SVM) [15–18] as a data multi-classification tool. This allows for more correct data transmission rate thanks to accurate classification of data. In other words, the accurate adjustment of data transmission rate in each wireless sensor node leads to the traffic balancing and congestion control. This improves the efficiency of the network.

SVM adjusts the transmission rate in sensor nodes in a distributed manner. Distributed systems only rely on peer to peer communication, and local decisions finally turn into a global one due to the spreading of local information [19–21]. SVM is a classification method with two main components: a kernel function and a set of support vectors. The support vectors are obtained via the training phase, based on the training data. New data are classified using a simple computation which involves only the kernel function and support vectors. SVM is used to solve various issues related to wireless sensor networks. In [22] authors focus on how to make SVM work over the sensor network in a fully distributed manner. They present a new method thereby training is performed only using one-hop communications between sensor nodes with low computation capability. This leads to reduced computational complexity associated with the data fusion process in both memory and computation, resulting in reduced the overall power requirements.

In this paper, we present a new method to adjust the data transmission rate, leading to congestion avoidance in WSNs. To the best of our knowledge, this is the first attempt to apply SVMs as a solution to a WSN congestion problem. We have chosen SVMs rather than other traditional classification methods such as Decision Tree, Naive Bayes classifier and nearest neighbor classifiers. This is because SVMs in this problem are able to yield better results than other methods. In addition, the application of this novel method precludes the problems emanating from many other methods (problems such as over fitting and the curse of dimensionality).

The rest of this paper is organized as follows: Section 2 describes the transmission rate adjustment strategy to control congestion in WSNs. Section 3 provides a brief background on SVM. The transmission rate based on SVM tuned by genetic algorithm (GA-SVM) is presented in Section 4. In Section 5, the performance evaluation is shown. Discussion about the results is presented in Section 6. This discussion shows the significant superiority of the proposed methods over the other methods, using Tukey test. Finally, Section 7 concludes the paper.

2. The transmission rate adjustment strategy

Bypassing the intermediate local hotspots cannot completely omit the congestion. In order to solve this problem, we propose a hop-by-hop congestion control scheme. In this method, each node receives the MAC layer channel information of downstream node and adjusts data transmission rate of the current node based on the traffic loading of two nodes (downstream node and current node). Clearly, if the current node has more traffic loading than the downstream node, there will be an increase in the packet transmission rate. In other cases, if the current node has less traffic than the upstream node, there will be a decrease in the packet transmission rate.

This strategy affects some metrics of network performance such as energy saving, transmission fairness, and network throughput. Each node sends an Awareness Packet (AP) to upstream node and informs it about its traffic information. When the current node processes the AP signal received from the downstream node, it should also consider its own congestion condition and adjusts the data transmission rate based on this information. Let us consider a packet (p) to be at node v (the current node), and this node is to forward it to the downstream node (i.e. node $\hat{v}$). To determine the traffic loading and data transmission rate in the current node, we define the following metrics:

2.1. Normalized queue length field

The queue length field at node v defines the number of packets in the node v:

$$V_q(v) = Q(v). \tag{1}$$

The normalized buffer size at node v (Q (v)) is defined as the buffer occupancy ratio of node v:

$$B_r(v) = \frac{Number\ of\ packets\ in\ the\ queue\ buffer}{Buffer\ size\ at\ Node\ V}. \tag{2}$$

The value of B_r (v) which denotes node traffic information is in the range of [0,1].

2.2. Congestion degree

Sometimes congestion happens because of the burst traffic. When a burst occurs around node v with low queue length, a lot of packets will enter queue v at the same time. Therefore, queue length is not solely a reasonable metric for recognizing the situation of traffic in each node. To cope with this issue, another metric named ‘congestion degree’ field is defined. Congestion degree indicates the changing tendency of the queue buffer in a period of time. The value of C_d is defined as follows:

$$C_d = \frac{T_s}{T_a} \tag{3}$$

In Eq. (3), T_a denotes the time interval between the arrivals of two adjacent data packets in MAC layer and T_s denotes the average processing time of data packets in the node. If $C_d > 1$, the arrival rate is greater than the departure rate of data packets. In other words, congestion may occur in the near future in this node. As a result, this node is not a suitable choice for the next relay node.

Assuming that the current node receives the AP signal successfully, in our local congestion processing method, R is the local node's data

Table 1
Comparison of various congestion methods in wireless sensor networks.

Protocol	Congestion Detection	Congestion Notification	Congestion Con troll	Congestion notification	H-by-H/ E-to-E	Application Type	Loss Recovery	Evaluation Parameters	Compared with	Classification approach
CODA [5]	Queue Length + Channel Load	Back-pressure message	AIMD	Explicit	H-by-H, E-to-E	Event	No	Energy Tax, Fidelity Penalty	No congestion control, open loop control	No
CCF2 [9]	Service rate, scheduling rate, buffer length	Information in header	AIMD method to Periodic Rate Control	Implicit	H-by-H	continuous	No	Good put, Fairness, Data Generation Rate, Link Layer Retrains-missions	alone	No
UHCC [10]	Packet delivery ratio, Buffer size	Information in header	Rate Control	Implicit	H-by-H	Periodic	No	Throughput, Fairness, Loss Ratio	PCCP, CCF	No
Prioritizing, For Qos [11]	End to End packet delay	Feedback message	Rate Control	Explicit	E-to-E	Event, periodic, continuous	Yes	Throughput, Data Loss, Priority Achieved	components of the protocol	No
FACC [12]	Channel busyness+, Queue length	Feedback message	AIMD method to Rate Control	Explicit	H-by-H	continuous	No	Packet Loss, Source Rate, Fairness, throughput	CODA, NCC	No
RCRT [13]	High Time to Re- pair Losses	New Rate in NACK header, or Feedback Rate msg	AIMD Rate Control	Explicit	E-to-E	All types	Yes	Good put, Rate, Packet, Reception	IFRC	No
ECODA [6]	Weighted Queue length, Delay dependent	Information in header	AIMD Rate adaptation	Implicit	H-by-H	Periodic	No	Throughput, E-to-E De- lay, Weighted Fairness	CODA	No
CADT [14]	Queue length, Link Capacity	Information in header	Rate adjustment	Implicit	H-by-H	Continuous	No	Packet Delivery Ratio, Packet, Delivery Latency, Queue Length	Alone	No
DRR [7]	Buffer Occupancy ratio	Feedback message	Rate adjustment	Explicit	H-by-H	Event	No	Packet delivery ratio, packet loss ratio, fairness index, average energy consumed	alone	No
Neural Network Congest-ion Detectio-n [8]	Buffer Occupancy ratio+ Traffic rate	Feedback message	–	Explicit	E-to-E	Continuous	No	Congestion level	alone	Yes
Our Work	Buffer Occupancy ratio, Congestion degree,	Information in header	Rate adjustment	Implicit	H-by-H	Continuous	Yes	packet loss, network throughput, energy consumed	CODA, ECODA, DRR	Yes

transmission rate, R_{max} is the maximum data transmission rate at each node, and ΔR determines the extent to which the amount of data transmission rate in each node is increased or decreased. Let B_{υ} be the buffer occupancy ratio of the current node (node υ) and $B_{\upsilon'}$ the buffer occupancy ratio of downstream node (node υ'). Both B_{υ} and $B_{\upsilon'}$ are within the AP signal. If $B_{\upsilon} > B_{\upsilon'}$ ($\Delta B > 0$), then the current node is more congested than the downstream node. Clearly, the local node should increase its data transmission rate. However, in this case, the congestion degree of an upstream node may be greater than the congestion degree of the current node ($C_{\upsilon'} > C_{\upsilon}$, i.e. $\Delta C < 0$). In other words, the downstream node has more tendency than the current node to change. In this situation, the algorithm decides to increase or decrease the transmission rate, using the trade-off between buffer occupancy and the congestion degree. In summary, when $\Delta B > 0$, $\Delta C > 0$ the algorithm should increase the data transmission rate. In contrast, when $\Delta B < 0$, $\Delta C < 0$ the algorithm should decrease the data transmission rate. If one of these variables has positive value and the other one has negative one, it is necessary to make a decision to increase or decrease the data transmission rate value, using an algorithm.

In this article, the SVM classification method is used to determine the exact amount of each node's data transmission rate based on traffic loading information (normalized queue length and congestion degree). In order to achieve this goal, the number of retransmission packets is determined based on different values of the buffer occupancy ratio (ΔB), congestion degree (ΔC) and data transmission rate (ΔR) in an assumed node. This information is given to SVMs. A support vector machine divides data into two categories: response variable and independent variables. The response variable is the number of retransmission packets, and the independent variables are ΔB, ΔC and ΔR. One SVM is designed for each response variable. For instance, data with zero retransmission value are specified with label 1 and other data are specified with label −1. Similarly, we obtain the support vector machine for every retransmission variable. Then, in each node the system derives the amount of retransmission packets based on the

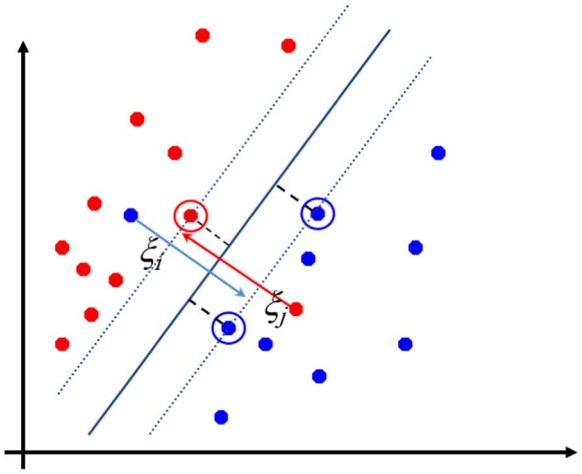

Fig. 1. Soft margin in SVM.

specified values of ΔB, ΔC and different values of ΔR. Finally, the transmission rate (ΔR) that produces less amount of retransmission value in accordance with existing ΔB and ΔC is selected as the final decision.

3. Support vector machine classification

Consider the problem of classifying data in a data space X into either one of two classes: G or ~G (not G). Suppose that each data point x has a feature vector $\vec{x}$ in some feature space $\vec{X} \in$ ^{n}. We are given k data points $x_1, x_2,\ldots, x_k$, called the "training points", with labels $y_1, y_2, \ldots, y_k$, respectively (where y_i=1 if $x_i \in G$ and −1 otherwise). We need to predict whether a new data point x is in G or not.

Support Vector Machine (SVM) is an efficient method to solve this problem. In the case of finite data space (e.g., the amount of data retransmission in each sensor network), the steps typically taken in SVM are as follows:

1. Define a kernel function K: $X \times X \rightarrow$. This function must be symmetric and the $k \times k$ matrix $[K(xi; xj)]_{i,j=1}^{k}$ must be positive semi-definite (i.e., has non-negative eigenvalues).
2.

$$Maximize \quad W(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} y_i y_j \alpha_i \alpha_j k(x_i, x_j). \tag{4}$$

3. *Subject to*

$$\sum_{i=1}^{k} y_i \alpha_i = 0 \tag{5}$$

$$0 \le \alpha_i \le C,\ i \in [i, k] \tag{6}$$

Suppose that $\{\alpha_1^*, \alpha_2^*, \ldots, \alpha_k^*\}$ is the solution of this optimization problem. We chose b=b^* so that $y_i h_K(x_i)$=1 for all i with $0 < \alpha_i^* < C$. The training points corresponding to such (i, α_i^*) so-called support vectors. The decision rule to classify a data point x is $x \in$ G if *sign* $(h_K(x))$ =1, where

$$h_k(x) = \sum_{i=1 \rightarrow k, x_i is\ a\ support\ vector} \alpha_i^* y_i (x, x_i) + b^* \tag{7}$$

According to Mercer's theorem [15], there exists a feature space $\vec{X}$ where the kernel K defined above is the inner product of $\vec{X}$ (i.e. K(x,z) = $<\vec{x}, \vec{z}>$ for every x,$z \in$ X). The function $h_k(.)$ represents hyperplane in $\vec{X}$, maximally separating the training points in X (G points on the positive side of the plane, ~G points in the negative side). It is probable that SVM has bounded classification error when applied to test data.

The allowance of softness in margins (i.e. a low cost setting) allows for errors to be made while fitting the model (support vectors) to the training/discovery data set. Sometimes it can be helpful to allow for errors in the training set, because it may produce a more generalizable model when applied to new datasets. In the case of SVM, an important assumption is that data are linearly separable. Yet this assumption is not correct in most cases. Initially Cortes and Vapnik [23] set non negative values as error values for each vector, aiming to find an optimal hyperplane separating two different classes. Based on the proposed method, Eqs. 4 to 6 are developed to deal with the optimization problem, where C denotes the margin regularization parameter whose function is to strike a balance between the maximization of margin width and the minimization of classifier errors. Should C be set with a big value, more attention would be directed to error. Such a definition of SVM is called soft-margin SVM [24]. Fig. 1 displays soft margin in SVM.

Some studies have been conducted to analyze the performance guarantee of SVM [25]. It can be assumed that if the data lie in a ball of radius B, and that the hypothesis set consists of hyperplanes of width (the margin), then the V C-dimension is $O\left(\frac{B^2}{\gamma^2}\right)$ [25].

4. Transmission rate adjustment based on GA-SVM

We consider a large number of sensor nodes deployed randomly in an area. Details of nodes distribution are described in Section 5. The algorithm is described as follows:

1) At first, we obtain the amount of retransmission based on different values of ΔB, ΔC and ΔR for several nodes.
2) We divide each data into two categories: independent variables and the response variable. The former consists of buffer occupancy ratio (ΔB), congestion degree (ΔC) and data transmission rate (ΔR) and the latter consists of the number of retransmission packets. So, the fourth dimension (retransmission) is interpreted by drawing on the first three dimensions.
3) **Train phase:** In this step, 80% of data are selected randomly as training data. Then, we rewrite training phase data by using SVMs to separate space. One SVM is designed for each response variable. For instance, at first, data with zero retransmission value are specified with label 1 and other data are specified with label −1. Similarly, we obtain the SVM for every retransmission variable. In this issue, we have 5 SVMs (for retransmission equal to 0,1,2,3 and more than 3).
4) **Test phase:** At this step, the remaining 20% of the data are sent to each of the designed SVMs. In this problem, we achieve 5 outputs for each data in testing phase. Each output shows whether the data belong to the respective support vector machine or not. Here, one of the following scenarios occurs for each data:

– First case: For a number of test data, only one of support vector machines returns the value 1, and other SVMs return −1. In this case, the data belong to a corresponding SVM that have been labeled 1.
– Second case: More than one Support Vector Machine returns the value 1. In other words, some SVMs claim ownership of data. In these circumstances, we calculate the average distance of data with these support vectors and assume that data belong to a category with greatest average distance from its support vector (for example data A in Fig. 2).
– Third case: Test data have been labeled −1 in the output of all support vector machines. In other words, the data do not belong to any classes. In this case, we calculate the average distance of data with all support vectors and assume that data belong to a category with minimum average distance from its support vector (for example data B in Fig. 2).

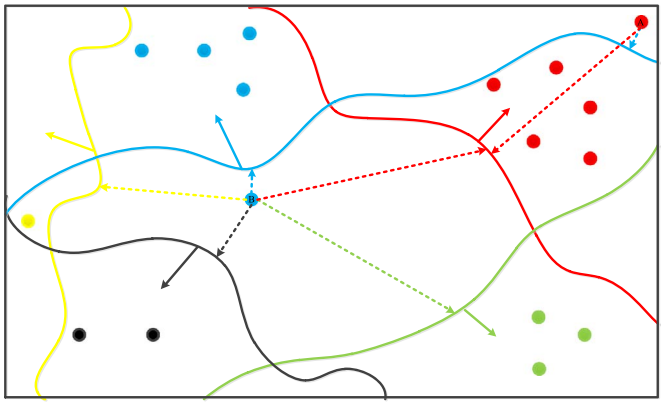

Fig. 2. Displays how data belong to various SVM classes.

Fig. 2 displays how data belong to various SVM classes. The functions shown in various colors represent the SVM classes, obtained from the training data. Five colors are used to represent the classes and there are five district classes. The color of data in each class is the same as the color of that function.

5) **Genetic Algorithm in Support Vector Machine:** The proper functioning of SVM-based classification depends on its parameters. As a result, each of the support vector machines must be used with appropriate parameters. In this study, the adjustable parameters are penalty ratio (C), acceptable error in SVM(ε) and the deviation of the Gaussian kernel function (δ). The above parameters are tuned simultaneously for all the support vector machines by using the genetic algorithm.The genetic algorithm is implemented, using uniform crossover and uniform mutation [25]. Fig. 3 shows how genetic algorithm is used to adjust the parameters of SVM.

As shown in the pseudo code, the preliminary responses in genetic algorithm were produced randomly. In various repetitions of genetic algorithm, the preliminary responses improved through uniform crossover and uniform mutation operators in order to satisfy stop condition [26].

6) **Action Phase:** At this Step, the system designed by GA-SVMs is given to all sensor nodes. Each node has certain values of ΔB, ΔC at any moment. The system has seven discrete selections (values between −3 to 3) for ΔR. Each node runs SVM seven times (the algorithm is executed once for each ΔR). In other words, each node obtains the retransmission value by using the specified values of ΔB, ΔC and each of seven values of ΔR, selecting a ΔR with a minimum value of the retransmission as the final decision. It should be noted that only the action phase is done online while other steps of the algorithm are taken offline. As a result, energy consumption is reasonable, using this method.

Fig. 4 displays a flowchart demonstrating the combination of SVM network (NSVM) with genetic algorithm. As Fig. 4 shows, the datasets are divided into test and train sets. Genetic algorithm operators obtain the appropriate values of SVM network parameters in train phase. Obviously, NSVM is formed, based on parameters obtained from GA and is used to estimate fitness values for each GA solution. These steps are repeated until the termination condition are met (50 iterations is the termination condition in this paper). Finally, the obtained parameters in the train phase are transferred to test phase where the percentage of classification error is estimated, using data in test phase.

Given their limited processing and memory ability, sensor nodes are not able to process complex operations. Put it simply, as these nodes have a short life time battery, processing complex operation and excessive use of sensors will lead to their death. Therefore, it is necessary to decrease the amount of processing performed by sensor nodes. This also holds true for wireless sensor network in the proposed congestion avoidance SVM method. In fact, in the proposed SVM method, training processes which require relatively high load of processing are performed at the workstation. Then, the designed SVMs are sent to the sensor nodes. The action phase which does not need a high level of processing is implemented in each node, using designed SVMs in distributed and online mode.

```
    Parameter Setting (number of iterations, Pop Size, Crossover Rate, Mutation Rate)
For i = 1 to number of Pop Size do
    i^ikh Population generated Randomly
    % (Generate uniformly distributed NSVM parameters between upper and lower bounds)

    i^ikh Population evaluated by the  fitness function
    % (Creating NSVM by the generated NSVM parameters in the i^ikh population and calculated error
    classification in the test data)
End For
For it = 1 to the number of iterations do
Childs= Pop Size * Crossover Rate
For i = 1 to number of Childs do
i^ikhChild is generated by the uniform crossover operator
  % (Parents are selected by the roulette wheel selection in the uniform crossover)

    i^ikhChild evaluated by the fitness function
    % (Creating NSVM by the generated NSVM parameters in the i^ikhchilds and calculated error classification
    in the test data)
End For
Mutation size = Pop Size * Mutation Rate
For i = 1 to number of Mutation Size do
i^ikhChild of mutation is generated by the uniform Mutation operator
  % (Parents are selected by the roulette wheel selection in the uniform Mutation)

    i^ikhChild of mutation evaluated by the fitness function
    % (Creating NSVM by the generated NSVM parameters in the i^ikh child of mutation and calculated error
    classification in the test data)
End For
Merging Population
% (Merge the Population, Childs of Crossover and Childs of Mutation)
Updating Best solution
End For
```

Fig. 3. Pseudo code that shows how genetic algorithm is used to adjust the parameters of SVM.

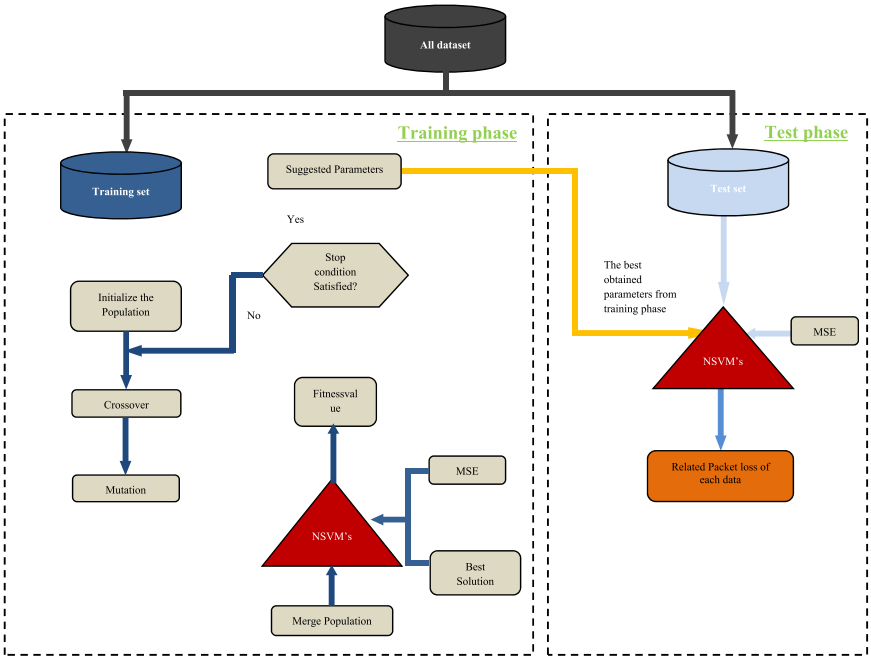

Fig. 4. Flowchart of SVM network.

5. Simulation results

Simulation results consist of two subsections: at first, the performance of SVM classification is evaluated. Additionally, the overall error of SVM method is compared with other classification techniques. Next, the efficiency of the proposed method is evaluated compared to three other hop-by-hop congestion control methods.

Matlab version 2013a and NS2 simulator [26] are two main softwares used to evaluate the efficiency of the proposed protocol. The Support Vector Machine is implemented using MATLAB and sensor network environment is implemented using NS2. These two softwares can be related as both of them use C++.

Test and training data were selected randomly out of 400 data sets. The table which is attached in Appendix A shows data details and how they are distributed in train and test sets. The values of packet loss in this table are calculated based on the values of ΔB, ΔC and ΔR using NS2 simulator in nodes.

Fig. 5 shows the simulation results. Axes X, Y, and Z represent ΔB, ΔC, and ΔR, respectively. Data involving zero packet loss are shown in red. Other data which have various degrees of packet loss are shown in other colors. Given that the packet loss greater than 4 is useless, all of these dates are shown in the same color (yellow). Moreover, the higher and lower bounds for each feature are recognizable through 3D axes in Fig. 5. For example, data transmission rate is between −4 and +4.

As Fig. 5 shows, since the data complexity is high, it is very difficult to draw a boundary between distinct classes. As a result, it is not easy to classify the data, making it necessary to use a powerful tool to classify data. SVM can be selected as such a powerful classification tool.

5.1. Efficiency of Support Vector Machine Classification

As mentioned earlier, total number of inputs are 400 data in the simulation, with training phase data and testing phase data making up 80% and 20% of total data, respectively.

As mentioned in the previous section, the SVM system parameters are regulated by using the genetic algorithm. The population size, crossover percentage, and mutation percentage are 50, 0.7, and 0.3, respectively. Designed chromosome in the genetic algorithm is as follows:

$$\begin{bmatrix} SVM1 & SVM2 & SVM3 & SVM4 & SVM5 \\ c_1 & c_2 & c_3 & c_4 & c_5 \\ \varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \varepsilon_4 & \varepsilon_5 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 \end{bmatrix}$$

Fig. 6 shows the convergence graph for the designed SVMs. In this figure, the horizontal axis and vertical axis represent the repetition of genetic algorithm and the degree of errors in test data, respectively.

The figure demonstrates that in the testing phase of the proposed algorithm, after 34 iterations, almost 77% of answers displayed in the output are accurate. The total number of iterations is 50 reps. Coefficients obtained by the genetic algorithm for five SVMs are shown Table 2.

Mean Square Error of each SVM to separate a specific category from other data is presented in Table 3.

The extent to which the results obtained by the SVM match the real data is displayed in Fig. 7. Blue and red lines represent the SVM training phase data and real data, respectively. Fig. 8 shows the compliance level of the data obtained by the SVM testing phase data and real data. As can be seen, most of red and blue lines overlap,

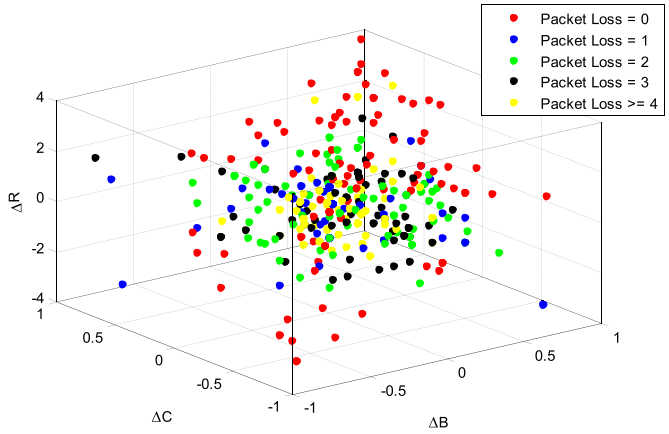

Fig. 5. Simulation result of packet loss with different input values.

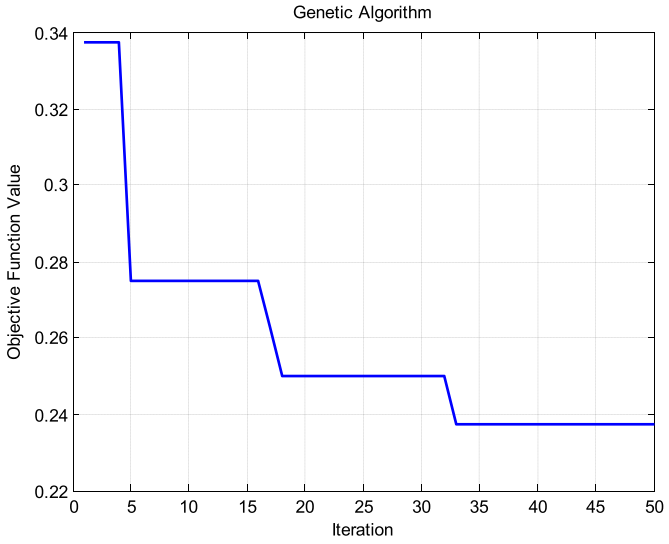

Fig. 6. The convergence graph support vector machines using genetic algorithms.

Table 2
Coefficients obtained for different support vector machines.

Parameters	SVM0	SVM1	SVM2	SVM3	SVM4
C	200	115.4864	95.8697	134.8977	68.7600
ε	0.0642	0.0732	0.0417	0.0340	0.0421
δ	0.3361	0.9130	1.2785	0.1717	1.1537

Table 3
Mean Square Error (MSE) for the test phase of each SVM to separate a specific category from other data.

SVM0	SVM1	SVM2	SVM3	SVM4
0.05	0.175	0.1625	0.1	0.1375

showing high quality of the presented classifier. It is worth noting that in Figs. 7 and 8, the horizontal and vertical axes represent the available data and the amount of packet loss, respectively.

In the field of machine learning, a confusion matrix [28], also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice-versa). Since the proposed algorithm has been built using five SVMs, the proposed matrix is a matrix of 5×5. The confusion matrix value of the suggested method with the eighty test data is presented below:

$$\begin{bmatrix} 18 & 2 & 0 & 0 & 1 \\ 1 & 7 & 1 & 0 & 0 \\ 0 & 2 & 14 & 2 & 1 \\ 0 & 0 & 5 & 7 & 1 \\ 0 & 2 & 1 & 0 & 15 \end{bmatrix}$$

All correct guesses are located on the diagonal of the matrix. Thus, it's easy to visually inspect the matrix for errors as they will be represented by values outside the diagonal. Upper triangular elements are the values which are predicted more than the real data by the classification system. In contrast, lower triangular elements are the values which are predicted less than real data by the classification system. As can be seen in the matrix, 77% of outputs are located on the diagonal. In other words, 77% of the data are predicted accurately. Besides diagonal data, upper triangular data in this problem have acceptable results because the upper triangular elements had expected

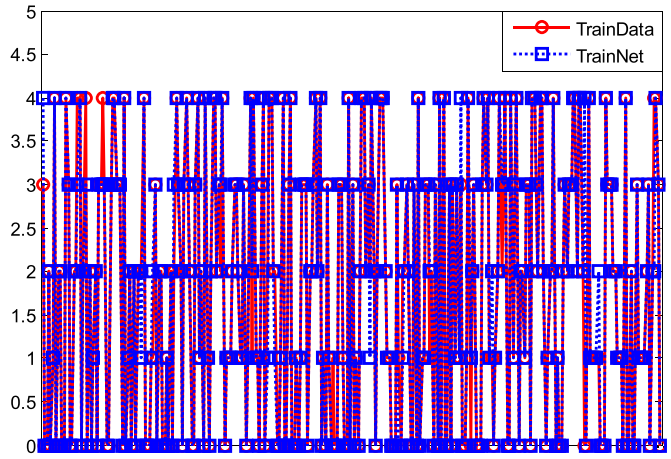

Fig. 7. Accordance level of real data and training data.

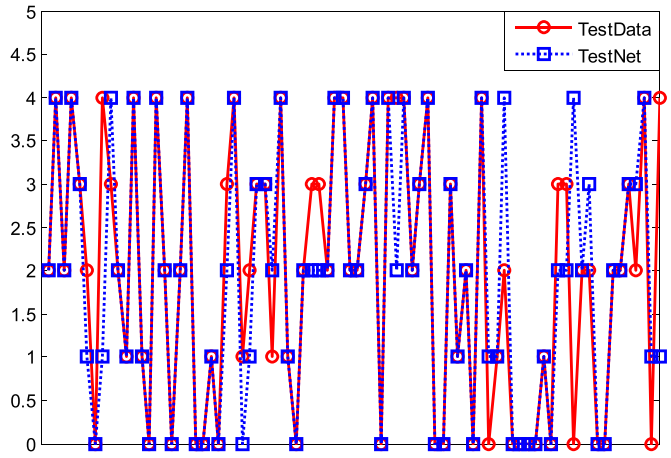

Fig. 8. Accordance level of real data and test data.

higher data transmission rate than the actual transmission rate in the node. As a result, we can conclude that the predicted values for the amount of retransmission are mostly correct, based on different values of the input variables.

5.1.1. Comparing the results of the SVM classifier with other methods of classification

To evaluate the quality of SVM classification, the overall error of this method is compared with other classification techniques. The training and testing data are assumed to be similar in all classification methods.

5.1.1.1. K-Nearest neighbor algorithm. the *k*-Nearest Neighbor algorithm [29] (or *k*-NN for short) is an optimization algorithm to find the closest point in metric spaces. The algorithms contain set S (a number of points in a metric space such as M) and appoint query q∈M. The goal is to find the closest points in set S to q. In other words, the algorithm returns K-nearest neighbors to the goal.

Given that the result of the *k*-Nearest Neighbor algorithm is dependent on the number of selected neighborhoods, this method is implemented on the basis of the number of different neighbors as well as several training and testing data. Simulation results are shown is Fig. 9.

The blue and red lines in Fig. 9 represent the training and testing errors based on different neighborhood values. As shown in Fig. 9, the minimum error for test data is 0.25 and it is obtained when the neighboring number is two. As shown in Fig. 9, the data errors in the

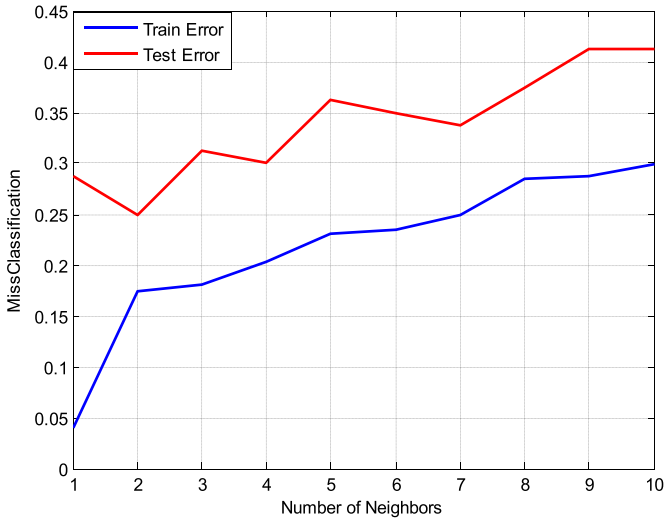

Fig. 9. The training and testing error in *k*-Nearest Neighbor algorithm based on various neighborhoods.

train phase are fewer than those in testing phase, with both types of errors increasing as the number of the neighbors increase.

5.1.1.2. Decision tree. A decision tree [30] uses the tree as a tool for decision modeling. Decision trees are typically used in operations research. However, it's used in decision analysis to determine the strategies that are most likely to reach a goal. The decision tree method has the following advantages compared to other classifiers:

1. The simple understanding: each person can learn how to work with decision tree by a little education.
2. Working with complex data: simple decision tree can work with complex data, making decisions on them easily.
3. The ability to combine with other techniques: the result of a decision tree can be combined with other techniques of decision-making to achieve better results. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. If in practice decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probabilistic model as a best choice model or online selection model algorithm. Decision trees are also used as a descriptive means for calculating conditional probabilities.

Fig. 10 shows the decision tree formation to solve the problem. The test data error of decision tree method to solve the problem is 0.3375.

5.1.1.3. Naive Bayes classifier. Naive Bayes [31] is a simple way to categorize phenomena, based on the probability of occurrence or non-occurrence of this phenomenon. Based on the intrinsic properties of the possibility (especially share probability), Naive method will provide good results after some initial training. Naïve uses a supervised method for learning. For example, a fruit may be apple, if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

The error rate of test data in Naive Bayes technique is 0.55.

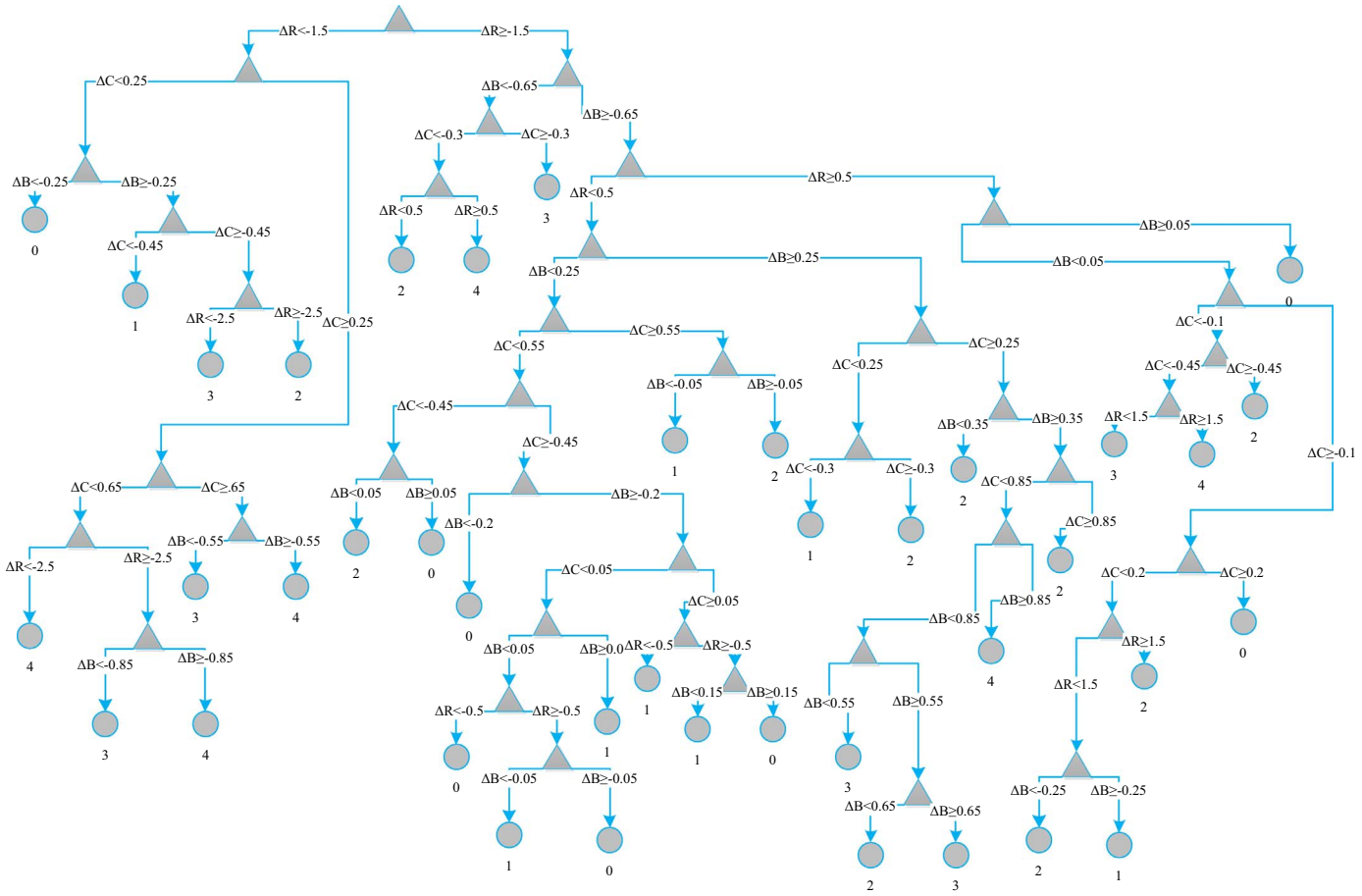

Fig. 10. The graph of binary decision tree results.

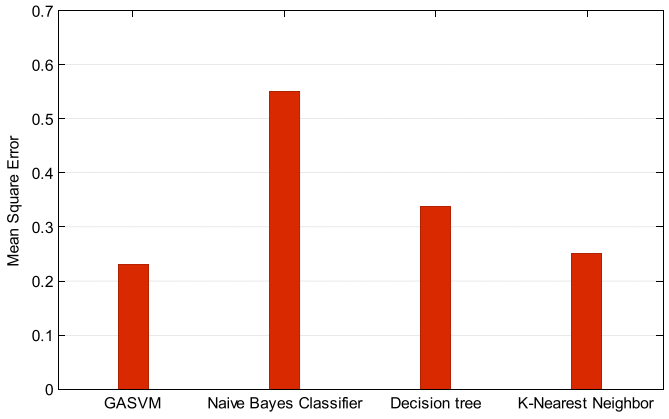

Fig. 11. Comparison of test data classification error in different classifiers.

SVM has been selected as the classification tool due to its lower classification error compared to other methods. Fig. 11 displays the amount of data classification errors in the graph.

As shown in Fig. 11, test data of SVM classifier has fewer errors than other methods of classification.

5.2. Discussion about for Performance Evaluation of GA-SVM Congestion Control

In this subsection, the performance of the proposed method is evaluated compared to three other hop-by-hop congestion control methods (i.e. CODA, ECODA, and DDR), using simulation experiments implemented by NS2 simulator [27]. Also, NS2 is used to obtain the values of the response variable (the amount of packet loss) on the basis of the different values of the input variables (ΔB, ΔC, and ΔR).

The following assumptions are made to simulate the sensor environment: There are 100 homogeneous sensor nodes (including source nodes and sink node), with the sink being at the upper right place. The IEEE 802.15.4 standard is used at MAC and physical layers. Each burst occurred between 120 and 160 s. There are two bursts for each event which have been shown by the blue marker in Fig. 12. A summary of the simulation parameters is displayed in Table 4.

Table 4
Simulation parameters.

Simulator name	NS-2 Version
Physical layer Protocol	IEEE.802.15.4
Beacon mode	Enable
CAP transmission Type	Direct, Using CSMA/CA
CFP transmission Type	GTS Transmission
Radio Type	Radio-Accnoise (Standard radio model)
Node Placement	Random
Application Type	Event Driven
Frequency band	2.4Ghz
Link layer transmission rate	250kbps
Range of Channel Connection Window size	[1,63]
Maximum CSMA/CA Attempts	4
Simulation Time	400 s
Area Size	100 m *100 m
Network Architecture	Homogeneous, Flat
Number of nodes	100
Largest Depth	20
Average Node degree	6
Sink Coordinate	(15,15) m
Communication Radius	5 m
Packet Size	25byte
Buffer Size	Packets 10

5.2.1. Network throughput

Network problems such as channel congestion and overflowed buffers can cause packet loss. Network throughput (receiving packet rate (RPR)) is defined as the rate at which the sink receives packets to the total packets sent via sources. The proposed algorithm predicts the right amount of data transmission rate. Therefore, congestion is managed properly and the receiving packet rate is improved. Fig. 12 depicts the rate at which the sink received packets in the proposed scheme compared with three other congestion control methods.

Obviously, when traffic load increases, a larger number of packets enter into the network. As a result, there is a possibility of congestion, which drops the packets. Practically, CODA dropped most of the packets in the burst places. The proposed method improves the packet delivery ratio by adjusting the transmission rate.

Previous experiments demonstrated the ability of our method to receive maximum packets in the sink in burst and heavy traffic environment. Now, we configure the light load environment to investigate the compatibility of our method to various loading environments. Fig. 13 shows that almost all methods are good during the light loading network. However, due to congestion near the sink node, the proposed method has a better receiving packet rate than the others.

5.2.2. Energy efficiency

The energy consumption in the nodes and lifetime are two important factors used to evaluate the energy efficiency of network. Energy consumption evaluates the medium energy consumption per bit needed to successfully transmit a packet to the sink, i.e., the ratio of the sum of consuming energies to send or transmit all packets at all nodes and total bits in data packets received at the sink. The energy

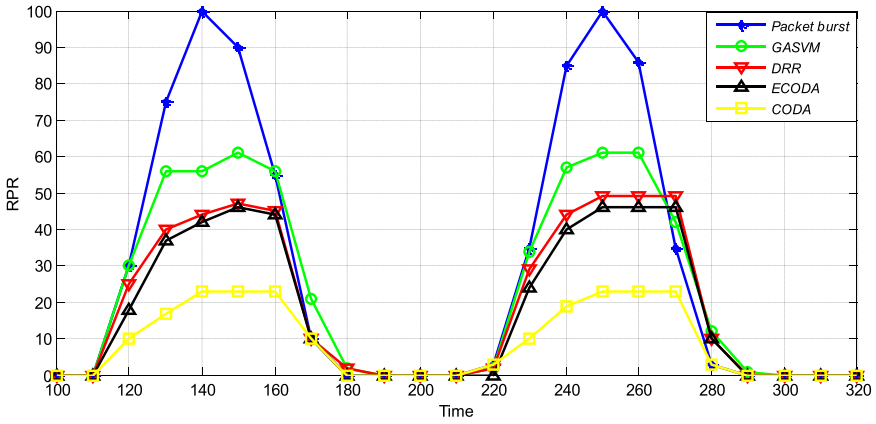

Fig. 12. Receiving packet rate under two bursts.

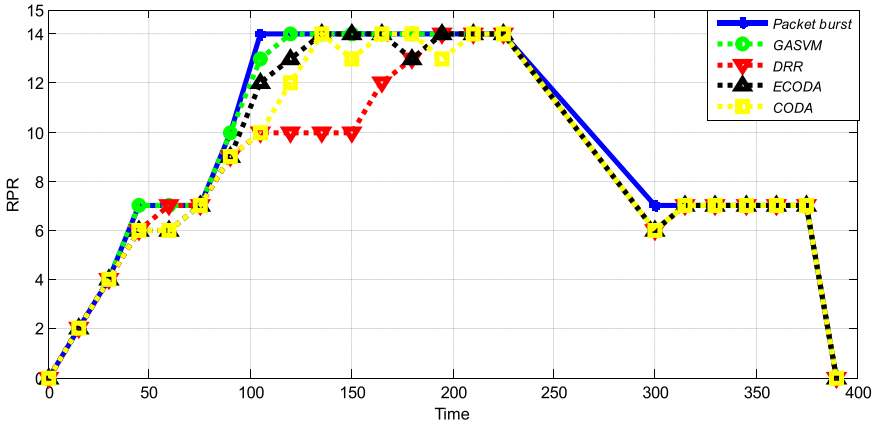

Fig. 13. Receiving packet rate under light load.

consumption in each node is made up of two parts: first part is the energy consumption due to processing at the sensor nodes and the second part is the energy consumption for communicating and receiving all packets, including data, acknowledgement (ACK), and AP packets. Table 5 shows the average energy consumption per bit, using different methods. The proposed method manages congestion better than other methods. Therefore, the amount of packets lost (due to packet collisions or full receiver buffer) is less than that in other methods. This results in less retransmission, energy saving as well as avoidance of energy wasting.

Network lifetime represents the duration in which the algorithm is run until the first node dies. As Table 6 shows, GSVM involves the longest network lifetime compared to other methods, because it can manage the traffic better that other methods through exact adjustment of the transmission rate. Therefore, the number of packets lost in the proposed method is less than that by other methods, leading to reduced packet retransmission in nodes.

5.2.3. Normalized buffer size and end to end delay

Fig. 14 shows the normalized buffer size $Q(v)$ over time for the nodes engaged in the packets transfer. The mean size of normalized buffer is a tool that is essentially used to measure the delay. The higher normalized buffer leads to an increase in delay. This is because when the normalized buffer size increases, the collision and overflow of packets are more likely to happen, leading to resending and the delay in the transfer of the packets. As Fig. 14 shows, all methods involve a bigger normalized buffer size at burst time (interval of 120–160 s and 240–280 s) compared to other times. The mean amount of normalized buffer size in the proposed methods is in a blue line, showing better results compared to other methods. The appropriate adjustment of transmission rate in the nodes and sending the suitable amount of packets to the next node are the two main factors contributing to this good performance.

The average end to end delay is another essential parameter evaluated in GASVM. End to end delay is defined as the time required send a packet from the source to the sink, including the transfer duration, the processing in relay nodes, and the resending of the packets as they may be lost. Delay parameter is considered as one of the essential issues in real-time applications. Fig. 15 shows that the amount of delay using GASVM is the least compared to other methods. The correct transmission rate adjustment leads to a decrease in lost packets and hence a decrease in resending of packets. This, in turn, leads to decreased delay.

Table 5
Comparison of energy consumption percentage between Methods.

Method	CODA	ECODA	DDR	SVM
Average energy consumption %	98.2	75.6	72.3	60.5

Table 6
The average values of network lifetime in different methods.

Method	CODA	ECODA	DDR	SVM
Second	198	221	239	292

6. Discussion

We used Tukey test to compare algorithms performance. Tukey test [32] is a more general form of t-test used to compare the receiving packet rate of various algorithms. In variance analysis tests, the independent variable or grouping variable has more than two levels, with dependent (output) variable being a quantitative one. This test is used to examine the difference between means of multiple groups (more than 2 groups or inputs).

To compare these algorithms receiving packet rate, we used a hypothesis test presented in Eq.8. If the ANOVA test is significant, then the Tukey test is used for algorithms grouping. ANOVA test only shows the difference among the means. It doesn't indicate which group has a mean, different from other groups. Using the Tukey test, these differences can be identified.

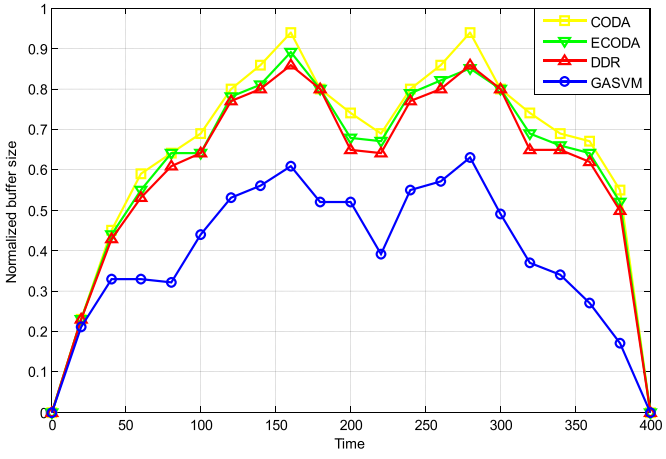

Fig. 14. The normalized buffer size $Q(v)$ over time.

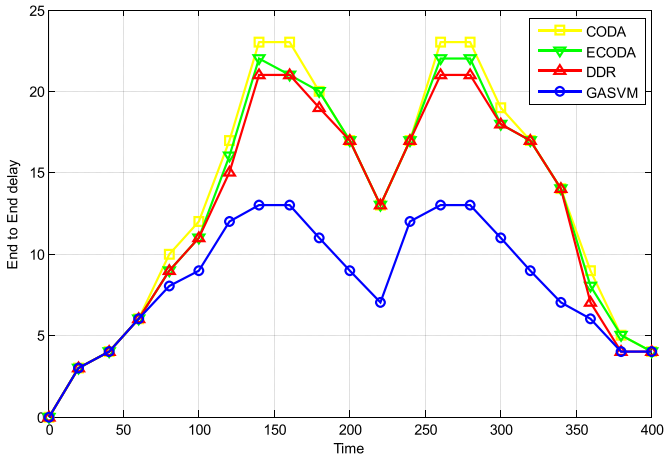

Fig. 15. The average end to end delay.

$$\begin{cases} H_0: \mu_{GASVM} = \mu_{DDR} = \mu_{ECODA} = \mu_{CODA} \\ H_1: Other\ wise \end{cases} \quad (8)$$

As ANOVA Table 6 shows, there is a significant difference among the methods. As a result, the H_0hypothesis is rejected. To have a better analysis, Tukey test was used in the burst times (simulation 120 s and 160 s).The Tukey-test results are shown in Table 7. The receiving packet rate at the burst time was adopted as the ranking criterion in this table.

Table 8 represents algorithms grouping where GASVM belongs to group A. DDR and ECODA belong to group B and CODA belongs to group C. It means that difference between mean of GASVM and those of other algorithms is significant. Furthermore, the difference between the mean of DDR and that of CODA is significant. Moreover, the difference between mean of ECODA and that of CODA is significant. But the difference between the mean of DDR and that of ECODA isn't significant.

As the Fig. 16 shows, there is a significant difference between receiving the packet rate at bursting time obtained from the proposed method and those obtained from other methods. These results prove that GASVM algorithm is preferred at a confidence level of 95%. Now by checking the Box-Plots of algorithms, presented in Fig. 17 it is clear that GASVM works better than other algorithms. Box plot shows the results of all runs.

Table 7
One-way ANOVA to compare Methods.

Source	DF	SS	MS	F	P-value	Result
Factor	3	24416.8	8138.9	169.42	0.000	Rejected
Error	160	7686.5	48.0			
Total	163	32103.2				

Table 8
Grouping information using Tukey test.

Algorithm	N	Mean	StDev	Grouping	Rank
GASVM	41	53.976	8.748	A	1
DDR	41	40.366	6.020	B	2
ECODA	41	38.366	7.921	B	3
CODA	41	19.707	4.082	C	4

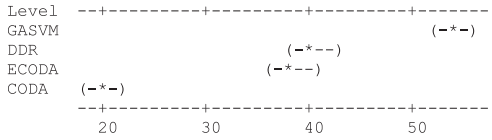

Fig. 16. Individual 95% CIs for Mean Based on Pooled StDev.

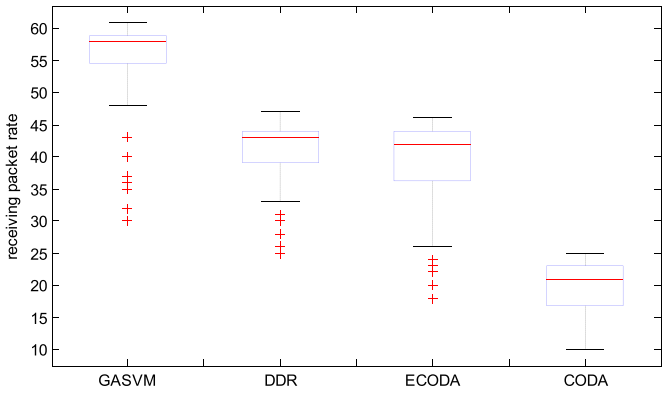

Fig. 17. Box-plot for evaluating statistical results graphically.

7. Conclusion

Many studies have alluded to the capability of various methods in classification, with a focus on the applications of SVM. Also, different researchers have taken different approaches to the investigation of the congestion problem in WSNs. However, no study has examined the efficiency of classification-based methods. This study aims to investigate the role of such methods in solving congestion problems in WSNs, using multiple classification methods. Indeed, this study develops a method, combining SVM network and genetic algorithm.

This algorithm avoids congestion while increases the throughput and the number of packets delivered to the sink node. Additionally, this approach decreases energy consumption at the nodes as well as the number of collisions at intermediate nodes.

This method adjusts the data transmission rate of each node by taking into consideration the traffic in the current and downstream nodes. To achieve this goal, a classification method by SVMs is presented. This method obtains the retransmission value by using the specified values of the buffer occupancy ratio, congestion degree and each of the different values of the transmission rate. Afterwards, each node selects a transmission rate with a minimum value of the retransmissions as the final decision. A genetic algorithm is used for tuning SVMs parameters.

Simulations show that SVM is the best tool for data classification. In addition, simulation demonstrates that the proposed method results in higher throughput and less energy consumption. Tukey test was used to demonstrate the significance superiority of the proposed method over other methods. The superiority of the solution proposed by this paper, compared to other classification methods (e.g. k-Nearest Neighbor algorithm, decision tree and Naive Bayes Classifier) is due to its efficiency in dealing with the complex data. As discussed in the paper, the data complexity contributes to the difficulty of data classification. The proposed SVM in this paper has adapted with the data complexity, thanks to adjusting its parameters by genetic algorithms.

Appendix A. Data details and how they are distributed in train and test sets

o	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT
1	−0.4	−0.8	−2	0	Test	51	0	−0.7	0	2	Train	101	−0.7	−0.6	−3	0	Train	151	0.5	0	0	2	Train
2	0.8	−0.4	−2	2	Test	52	0	0.8	0	2	Train	102	−0.9	−0.9	−3	0	Train	152	0.7	0	0	2	Train
3	−0.2	0.9	1	0	Train	53	0	0	−1	1	Train	103	−0.8	0.7	−3	1	Train	153	0.8	0	−1	3	Test
4	0.9	0.3	−1	4	Test	54	0	0	1	1	Train	104	−0.5	−0.6	−2	0	Train	154	0.6	0	1	0	Test
5	−0.7	−0.5	1	4	Train	55	0	0.6	1	0	Train	105	−0.4	−0.4	−1	0	Train	155	0	0.3	0	0	Train
6	0.8	−0.2	1	0	Train	56	0.2	0	1	0	Train	106	−0.3	−0.3	−1	0	Train	156	0	0.5	0	1	Train
7	−0.2	0.8	1	0	Train	57	0.7	0.3	3	0	Test	107	−0.1	−0.1	−1	0	Train	157	0	0.7	0	2	Train
8	0.3	0.5	1	0	Train	58	0.5	0.4	0	3	Train	108	0.8	0.8	3	0	Train	158	0	0.9	0	3	Test
9	0.8	0.8	3	0	Test	59	0.6	0	2	0	Train	109	0.7	0.5	2	0	Train	159	0	0.3	0	1	Train
10	0.9	0.7	3	0	Test	60	0.5	0	0	2	Train	110	0.5	0.1	2	0	Train	160	0	0	0	0	Train
11	0	−0.5	0	2	Train	61	−0.7	−0.9	3	4	Train	111	0.3	0.2	1	0	Train	161	−0.1	−0.9	−1	2	Train
12	0	0.6	0	2	Test	62	0.7	0.9	−2	4	Train	112	0.1	0.1	1	0	Train	162	0.1	−0.7	−1	0	Train
13	0	0	0	0	Test	63	−0.5	−0.7	−3	0	Train	113	0.1	0.8	2	0	Train	163	−0.1	0.7	−1	1	Train
14	0	0	1	1	Train	64	0.5	−0.5	−1	1	Train	114	0.2	0.7	2	0	Test	164	0.7	−0.1	−1	2	Train
15	0	0	0	0	Train	65	0.5	−0.5	1	0	Test	115	0.8	0.1	3	0	Train	165	−0.1	−0.6	1	4	Train
16	0	0.3	1	0	Train	66	0.5	−0.3	1	0	Test	116	0.7	0.2	3	0	Test	166	0.1	−0.5	1	2	Train
17	0	0.6	−1	2	Train	67	0.1	−0.8	−1	0	Train	117	0.1	0.1	0	0	Train	167	−0.1	0.7	1	0	Test
18	0	0.9	−2	4	Test	68	0.4	0.6	2	0	Train	118	0.1	0.1	0	0	Test	168	0.7	−0.1	1	0	Test
19	0	0.7	−2	3	Train	69	−0.2	0	2	1	Train	119	0.2	0.1	1	0	Train	169	−0.1	0.1	1	1	Train
20	0.1	0.1	0	1	Test	70	0.4	0	0	1	Train	120	0.1	0.2	0	0	Train	170	−0.1	0.1	−1	0	Train
21	−0.1	−0.1	−1	0	Train	71	0	−0.1	0	0	Test	121	−0.8	−0.7	3	4	Train	171	0.2	−0.2	0	1	Train
22	0.8	−0.8	1	0	Train	72	0	0.4	0	1	Train	122	−0.4	−0.4	1	3	Test	172	0.2	−0.2	1	0	Train
23	−0.8	0.8	1	1	Test	73	0	0	−1	2	Test	123	−0.6	−0.6	2	3	Train	173	0.2	−0.2	−1	1	Test
24	0.5	0.5	−1	3	Test	74	0	0	1	1	Train	124	−0.5	−0.5	1	3	Train	174	0.3	−0.3	1	0	Train
25	−0.7	−0.7	1	4	Train	75	0.4	0.2	1	0	Test	125	−0.5	−0.5	3	4	Test	175	0.2	−0.7	−1	0	Train
26	0.3	−0.3	1	0	Train	76	0	0.8	0	2	Train	126	0.5	−0.2	1	1	Test	176	0.2	0.8	0	2	Train
27	0.3	−0.3	3	1	Train	77	0.1	0	0	1	Train	127	−0.9	0.8	2	3	Train	177	0.8	0.2	0	2	Train
28	0.8	0.2	3	0	Train	78	0.4	0.5	0	3	Train	128	−0.1	−0.1	0	1	Train	178	0	0	2	2	Train
29	0.8	0.2	3	0	Train	79	0.1	0.1	0	1	Train	129	−0.1	−0.1	0	1	Train	179	0.5	−0.5	0	1	Train
30	0.6	0	0	2	Train	80	0.2	0.1	0	0	Train	130	0.8	0.6	−2	3	Train	180	0.5	−0.5	−1	2	Train
31	0.6	0	1	1	Train	81	−0.7	−0.5	−1	1	Train	131	0.9	0.7	−2	4	Train	181	0.5	−0.5	1	0	Train
32	0	0.8	0	2	Train	82	0.6	−0.1	−1	2	Test	132	0.7	0.5	−2	3	Test	182	0.2	−0.7	−1	0	Train
33	0	0.3	−1	2	Train	83	−0.6	0.1	−1	0	Train	133	0.4	0.3	−1	3	Train	183	0.2	−0.7	1	1	Train
34	0	0.6	2	0	Train	84	0.5	0.3	−3	4	Train	134	0.2	0.1	−1	2	Train	184	0.2	−0.7	0	1	Train
35	0	0	0	0	Train	85	−0.6	−0.7	1	4	Test	135	0.1	0.2	−1	1	Train	185	−0.7	0.2	1	2	Train
36	0	0	0	0	Test	86	0.6	−0.6	1	0	Train	136	0.3	0.1	0	2	Train	186	−0.7	0.2	−1	0	Train
37	0	0	0	0	Train	87	−0.5	0.5	1	2	Train	137	0.1	0.3	0	2	Test	187	−0.7	0.2	0	1	Train
38	0	0.2	0	1	Train	88	0.9	0.9	4	0	Train	138	0.6	0.1	−2	2	Train	188	−0.7	−0.1	1	3	Train
39	0.2	0	0	1	Test	89	−0.7	0	0	3	Train	139	0.5	0.1	−2	3	Train	189	−0.1	0.1	0	1	Train
40	0	0	0	0	Train	90	0.5	0	0	1	Train	140	0	0	0	0	Train	190	−0.1	0.1	1	1	Train
41	−0.5	−0.3	−3	0	Train	91	0	−0.1	0	0	Train	141	−0.1	0	−1	0	Test	191	−0.1	0.1	−1	0	Test
42	0.5	−0.5	−1	1	Train	92	0.3	0.6	−2	4	Train	142	0.1	0	1	0	Train	192	0	0.1	1	1	Train
43	−0.5	0.5	−1	0	Train	93	0	0	−1	1	Train	143	−0.7	0	−2	0	Train	193	0	0	−1	1	Train
44	0.6	0.7	−1	4	Train	94	0	0	1	1	Train	144	0	0.1	−1	1	Train	194	0	0	1	1	Train
45	−0.7	−0.7	2	4	Train	95	0.1	0.1	0	1	Test	145	0	−0.1	0	0	Train	195	0	0	0	1	Train
46	0.5	−0.4	1	0	Train	96	0.5	0.5	3	0	Train	146	0	−0.7	0	2	Train	196	0	0	0	0	Train
47	−0.1	−0.8	1	3	Train	97	0.4	0.5	2	0	Train	147	−0.7	0	0	3	Train	197	0.5	0.6	2	0	Train
48	0.7	0.1	2	0	Train	98	0.6	0.4	3	0	Train	148	−0.4	0	1	2	Train	198	0.5	0.6	−1	3	Train
49	−0.1	0.6	0	1	Train	99	0.5	−0.1	1	0	Train	149	−0.1	0	0	0	Train	199	0.5	0.6	0	2	Test
50	0.1	−0.1	1	0	Train	1	0.6	0.4	2	0	Train	150	0.6	0	0	2	Train	200	0.3	0.4	0	2	Train

	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT	No	ΔB	ΔC	ΔR	RT	DT
201	−0.4	−0.8	2	4	Train	251	−0.4	−0.7	0	3	Test	301	−0.8	−0.8	3	4	Train	351	−0.4	−0.7	2	4	Train
202	0.8	0.4	−2	3	Train	252	0.2	0.8	0	3	Train	302	0.8	0.8	−2	4	Train	352	0.2	0.8	1	1	Train
203	−0.2	−0.9	1	3	Test	253	0	0	−1	1	Train	303	−0.2	−0.9	1	3	Train	353	0	0	−3	2	Train
204	0.9	0.3	−2	4	Train	254	0	0	1	1	Train	304	0.9	0.3	−3	4	Train	354	0	0	3	2	Train
205	−0.7	−0.5	2	4	Train	255	0.7	0.6	−1	3	Train	305	−0.7	−0.5	1	3	Train	355	0.9	0.8	−1	4	Train
206	0.8	0.2	−1	2	Test	256	0.7	0.6	−2	4	Train	306	0.8	0.2	−2	3	Train	356	0.6	0.1	−2	3	Train
207	−0.2	−0.8	1	3	Test	257	0.7	0.6	−3	4	Train	307	−0.2	−0.8	3	4	Test	357	0.5	0.1	−3	3	Train
208	0.8	0.5	3	4	Train	258	0.5	0.4	0	3	Train	308	0.8	0.5	1	3	Test	358	0.5	0.1	0	2	Train

209	0.8	0.8	−3	4	Train	259	0.6	0.5	2	3	Train	309	0.8	0.8	−1	3	Train	359	0.6	0.9	2	4	Test
210	0.9	0.7	−3	4	Train	260	0.6	0.5	−3	4	Test	310	0.9	0.7	−1	3	Train	360	0.5	0.1	−3	3	Train
211	0.8	0.5	0	4	Train	261	−0.7	−0.9	3	4	Train	311	0.7	0.1	0	3	Train	361	−0.1	−0.2	3	2	Train
212	0.7	0.6	0	3	Train	262	0.7	0.9	−2	4	Test	312	0.7	0.3	0	3	Train	362	0.3	0.1	−2	1	Train
213	0	0	0	0	Train	263	−0.6	−0.7	3	4	Test	313	0.4	0.3	−1	2	Test	363	−0.6	−0.7	0	1	Test
214	0	0	1	1	Train	264	0.5	0.5	−3	4	Train	314	−0.4	−0.3	1	2	Train	364	0.5	0.1	−3	3	Train
215	0	0	0	0	Test	265	0.5	0.5	−1	3	Train	315	0.3	0.7	−1	2	Test	365	0.5	0.1	−1	2	Train
216	0.9	0.3	−2	4	Train	266	0.5	0.3	1	0	Train	316	0.9	0	−2	2	Train	366	0.1	0.3	1	0	Train
217	0.8	0.6	−1	3	Train	267	0.6	0.8	−1	3	Train	317	0.8	0.1	−1	2	Train	367	0.1	0.8	−1	2	Train
218	0.7	0.9	−2	4	Train	268	0.6	0.8	−2	4	Train	318	0.7	0.2	−2	2	Test	368	0.1	0.8	−2	2	Test
219	0.6	0.7	−2	4	Train	269	0.6	0.8	−3	4	Train	319	0.6	0.2	−2	2	Train	369	0.2	0.8	−3	2	Train
220	0.1	0.1	0	1	Train	270	0.4	0.5	−3	4	Train	320	0.1	0.1	−2	2	Train	370	0.1	0.5	−3	3	Test
221	−0.1	−0.1	−1	0	Train	271	0.4	0.5	−2	3	Train	321	−0.1	−0.1	−1	0	Train	371	0.1	0.2	−2	2	Train
222	0.8	0.8	−1	3	Test	272	0.9	0.9	−3	4	Train	322	0.8	0.8	−3	4	Train	372	0.9	0.9	3	0	Train
223	0.8	0.8	2	4	Train	273	−0.9	−0.9	3	4	Train	323	0.8	0.8	2	4	Test	373	−0.9	−0.9	−3	0	Train
224	0.5	0.5	−1	3	Train	274	0	0	1	1	Train	324	0.5	0.1	−1	2	Test	374	0	0	3	2	Train
225	−0.7	−0.7	1	4	Test	275	0.4	−0.2	1	0	Train	325	−0.3	−0.4	1	2	Train	375	0.1	−0.1	1	0	Train
226	0.3	0.3	1	0	Test	276	0.6	0.8	0	2	Test	326	0.5	0.8	3	0	Train	376	0.2	0.3	0	1	Train
227	0.3	0.3	3	0	Train	277	0.1	0.5	0	2	Test	327	0.3	0.3	1	0	Train	377	0.1	0.3	0	1	Train
228	−0.8	0.2	3	3	Train	278	0.4	0.5	0	3	Train	328	−0.8	−0.8	3	4	Train	378	0.4	0.5	0	2	Train
229	0.8	0.8	−3	4	Train	279	0.1	0.1	0	1	Train	329	0.8	0.8	0	2	Train	379	0.1	0.1	1	0	Train
230	0.9	0.9	−3	4	Train	280	0.2	0.1	0	0	Train	330	0.9	0.9	0	2	Train	380	0.2	0.1	−3	3	Train
231	0.9	0.9	−2	4	Train	281	−0.7	−0.5	−3	0	Train	331	0.9	0.9	−2	4	Train	381	−0.7	−0.5	−3	0	Train
232	0.8	0.8	−1	3	Train	282	0.7	−0.9	−3	1	Train	332	0.9	0.9	−3	4	Train	382	0.7	−0.9	−3	1	Train
233	0.3	0.3	−1	2	Train	283	0.6	0.9	−3	4	Train	333	0.3	0.3	−1	2	Train	383	0.6	0.1	−3	3	Train
234	0.7	0.6	3	0	Train	284	0.5	0.3	−3	4	Train	334	0.7	0.1	0	2	Test	384	0.5	0.9	−3	4	Train
235	−0.7	−0.5	2	4	Train	285	−0.6	−0.7	1	4	Test	335	−0.7	−0.5	0	2	Train	385	−0.6	−0.7	3	4	Test
236	−0.7	−0.7	2	4	Train	286	0.6	0.6	−1	2	Train	336	−0.7	−0.7	3	4	Train	386	0.6	0.4	−1	2	Test
237	0	0	0	0	Train	287	0.6	0.6	−2	3	Train	337	0	0	0	0	Train	387	0.6	0.4	−2	3	Train
238	−0.5	−0.2	2	3	Train	288	0.9	0.9	−3	4	Test	338	−0.5	−0.2	0	2	Train	388	0.9	0.6	−3	4	Train
239	0.2	0.9	0	2	Train	289	−0.7	−0.9	3	4	Test	339	0.2	0.9	0	2	Train	389	−0.8	−0.9	3	4	Train
240	0.5	0.9	−2	3	Train	290	0.5	0.2	0	2	Train	340	0.5	0.9	0	2	Train	390	0.7	0.2	0	3	Train
241	0.9	0.3	−3	3	Train	291	−0.6	−0.1	0	2	Train	341	0.9	0.3	−1	2	Test	391	−0.6	−0.2	0	2	Test
242	0.8	0.8	−3	4	Train	292	0.3	0.6	−2	3	Train	342	0.8	0.8	0	2	Test	392	0.3	0.7	−2	3	Test
243	−0.8	−0.8	3	4	Test	293	0	0	−1	1	Test	343	−0.8	−0.8	0	2	Train	393	0	0	−1	1	Test
244	0.6	0.7	−3	4	Train	294	0	0	1	1	Train	344	0.6	0.7	0	2	Train	394	0	0	1	1	Train
245	−0.7	−0.7	2	4	Test	295	0.1	0.1	−3	3	Train	345	−0.7	−0.9	2	4	Train	395	0.4	0.1	−3	3	Train
246	0.5	0.4	−1	3	Train	296	0.5	0.5	2	0	Train	346	0.9	0.4	−1	4	Train	396	0.5	0.6	2	0	Train
247	−0.1	−0.8	1	3	Train	297	0.4	0.5	1	0	Test	347	−0.1	−0.7	1	3	Train	397	0.4	0.9	1	0	Train
248	0.7	0.9	−2	4	Train	298	0.6	0.4	3	0	Train	348	0.8	0.9	−3	4	Test	398	0.6	0.4	3	0	Train
249	−0.1	−0.6	0	2	Train	299	0.5	−0.1	1	0	Train	349	−0.1	−0.6	1	3	Train	399	0.6	−0.1	1	0	Train
250	0.5	0.5	−2	3	Train	300	0.6	0.4	−2	3	Train	350	0.5	0.5	−2	3	Test	400	0.7	0.8	−2	4	Train

References

[1] Priyank Garg, Reena Rani, A survey on wireless sensor networks routing algorithms, IJITKM Spec. Issue, ICFTEM (2014) 38–42.

[2] M. Gholipour, M.R. Meybodi, LA-Mobicast: a learning automata based Mobicast routing protocol for wireless sensor networks, Sens. Lett. 6.2 (2008) 305–311.

[3] Gholipour Majid, Abolfazl T. Haghighat, Mohammad R. Meybodi, Hop-by-Hop traffic-aware routing to congestion control in wireless sensor networks, EURASIP J. Wirel. Commun. Netw. 2015.1 (2015) 15.

[4] Kaur, Jasleen, Rubal Grewal, and Kamaljit Singh Saini. "A survey on recent congestion control schemes in wireless sensor network." Advance Computing Conference (IACC), 2015 IEEE International. IEEE, 2015

[5] Wan, Chieh-Yih, Shane B. Eisenman, and Andrew T. Campbell. "CODA: congestion detection and avoidance in sensor networks." Proceedings of the 1st international conference on Embedded networked sensor systems. ACM, 2003

[6] Li. Qiang Tao, Feng Qi Yu, ECODA: enhanced congestion detection and avoidance for multiple class of traffic in sensor networks, Consum. Electron., IEEE Trans. on 56.3 (2010) 1387–1394.

[7] Deshpande, Vivek S., et al. "Congestion Control in Wireless Sensor Networks by using Differed Reporting Rate." Information and Communication Technologies (WICT), 2012 World Congress on. IEEE, 2012.

[8] P. Singhal, A. Yadav. "Congestion Detection in Wireless Sensor Networks using Neural Networks." International Conference for Convergence of Technology , April 2014 IEEE International. IEEE, 2014

[9] S. Brahma, M. Chatterjee, and K. Kwiat, "Congestion control and fairness in wireless sensor networks," in Proc. 8th IEEE Int. Conf. on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010, pp. 413–418.

[10] G. Wang, K. Liu, "Upstream hop-by-hop congestion control in wireless sensor networks," in Proc. IEEE 20th Int. Symp. on Personal, Indoor and Mobile Radio Communications, 13-16 Sept 2009, pp. 1406–1410

[11] A. Sharif, V. Potdar, A. J. D. Rathnayaka, "Prioritizing Information for Achieving QoS Control in WSN," in Proc. 24th IEEE Int. Conf. Advanced Information Networking and Applications (AINA), April 2010, pp. 835–842

[12] X. Yin, X. Zhou, R. Huang, Y. Fang, S. Li, A Fairness-Aware Congestion Control Scheme in Wireless Sensor Networks, IEEE Trans. Veh. Technol. 58 (9) (2009) 5225–5234.

[13] J. Paek, R. Govindan, RCRT: rate-controlled reliable transport protocol for wireless sensor networks (1–20:45, oct)ACM Trans. Sen. Netw. 7 (3) (2010) 20 (1–20:45, oct).

[14] M. Rahman, M. Monowar, and C. Hong, "A Capacity Aware Data Transport Protocol for Wireless Sensor Network," in Proc. Int. Conf. on Computational Science and Its Applications, ICCSA, ser. Lecture Notes in Computer Science, 2009, pp. 491–502

[15] Weston, Jason. "Support Vector Machine." Tutorial ⟨http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.Pdf⟩, accessed 10 (2014)

[16] Mengyu Fu, Yang Tian, Fang Wu, Step-wise support vector machines for classification of overlapping samples, Neurocomputing 155 (2015) 159–166.

[17] Yudong Zhang, et al., Magnetic resonance brain image classification via stationary wavelet transform and generalized eigenvalue proximal support vector machine, J. Med. Imaging Health Inform. 5.7 (2015) 1395–1403.

[18] Shuihua Wang, et al., Identification of green, oolong and black teas in China via wavelet packet entropy and fuzzy support vector machine, Entropy 17.10 (2015) 6663–6682.

[19] Shuai Li, Yuesheng Lou, Bo Liu, Bluetooth aided mobile phone localization: a nonlinear neural circuit approach, ACM Trans. Embed. Comput. Syst. (TECS) 13.4 (2014) 78.
[20] Shuai Li, Zheng Wang, Yangming Li, Using laplacian eigenmap as heuristic information to solve nonlinear constraints defined on a graph and its application in distributed range-free localization of wireless sensor networks, Neural Process. Lett. 37.3 (2013) 411–424.
[21] Shuai Li, Feng Qin, A dynamic neural network approach for solving nonlinear inequalities defined on a graph and its application to distributed, routing-free, range-free localization of WSNs, Neurocomputing 117 (2013) 72–80.
[22] Woojin Kim, et al., A distributed support vector machine learning over wireless sensor networks, Cybern., IEEE Trans. on 45.11 (2015) 2599–2611.
[23] V.N. Vapnik, Statistical Learning Theory, Wiley-Interscience, 1998.
[24] Shigeo Abe, Support Vector Machines for Pattern Classification 53, Springer, London, 2005.
[25] Paul, Saurabh, Malik Magdon-Ismail, and Petros Drineas. "Feature Selection for Linear SVM with Provable Guarantees." AISTATS. 2015
[26] Spector, Lee, Thomas Helmuth. "Uniform linear transformation with repair and alternation in genetic programming." Genetic Programming Theory and Practice XI. Springer New York, 2014. 137-153
[27] Teerawat Issariyakul, Ekram Hossain, Introduction to network simulator NS2, Springer Science & Business Media, 2011.
[28] Hempel, Susanne, et al. "Machine Learning Confusion Matrix, Text Terms Distinguishing Relevant and Irrelevant Citations, and Reviewer Disagreements." (2012)
[29] Salvador Garcia, et al., Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34.3 (2012) 417–435.
[30] Seokho Kang, Sungzoon Cho, Pilsung Kang, Constructing a multi-class classifier using one-against-one approach with different binary classifiers, Neurocomputing 149 (2015) 677–682.
[31] Michał. Woźniak, Manuel Graña, Emilio Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[32] John W. Tukey, Comparing individual means in the analysis of variance, Biometrics (1949) 99–114.

Majid Gholipour completed his B.S. and M.S. studies in Computer Engineering in Iran, dated 2002 and 2005 respectively. He is currently a Ph.D. candidate in Computer Engineering from Islamic Azad University (Tehran Science and Research branch). He joined the faculty of Computer Engineering Department at Islamic Azad University (Qazvin Branch) in 2005. His research interests include Wireless Sensor Networks, Cognitive Networks, Learning Systems, Traffic Engineering, and Soft Computing.

Abolfazl Toroghi Haghighat received his B.S. and M.S. degrees in Electrical Engineering from Sharif University of Technology, Tehran, Iran, in 1993 and 1996, respectively. His Ph.D. degree from Amirkabir University of Technology (AUT), Tehran, Iran, in 2003. He is an Assistant Professor of Computer Engineering and Information Technology at Islamic Azad University of Qazvin. His research interests are in wireless networks, pattern recognition, fault-tolerant computing, and distributed systems.

Mohammad Reza Meybodi received B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran in 1973 and 1977, respectively. He also received M.S. and Ph.D. degrees in Computer Science from Oklahoma University, USA, in 1980 and 1983, respectively. Currently, he is Full Professor in the Computer Engineering Department of Amirkabir University of Technology, in Tehran, Iran. Prior to his current position, he worked from 1983 to 1985 as Assistant Professor at Western Michigan University, and from 1985 to 1991 as Associate Professor at Ohio University, USA. His research interests include Channel Management in Cellular Networks, Learning Systems, Parallel Algorithms, Soft Computing and Software Development.