14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

TRAINING A SVM-BASED CLASSIFIER IN DISTRIBUTED SENSOR NETWORKS

K. Flouri[1], *B. Beferull-Lozano*[2], *and P. Tsakalides*[1]

[1]Department of Computer Science
University of Crete and
Institute of Computer Science (FORTH-ICS)
71110 Heraklion, Crete, Greece
{flouri,tsakalid}@ics.forth.gr

[2] Instituto de Robótica
Escuela Técnica Superior de Ingeniería (ETSI)
Universidad de Valencia (UV)
46071, Valencia, Spain
baltasar.beferull@uv.es

ABSTRACT

The emergence of smart low-power devices (motes), which have micro-sensing, on-board processing, and wireless communication capabilities, has impelled research in distributed and on-line learning under communication constraints. In this paper, we show how to perform a classification task in a wireless sensor network using distributed algorithms for Support Vector Machines (SVMs), taking advantage of the sparse representation that SVMs provide for the decision boundaries. We present two energy-efficient algorithms that involve a distributed incremental learning for the training of a SVM in a wireless sensor network, both for stationary and non-stationary sample data (concept drift). Through analytical studies and simulation experiments, we show that the two proposed algorithms exhibit similar performance to the traditional centralized SVM training methods, while being much more efficient in terms of energy cost.

1. INTRODUCTION

One of the most important tasks to be performed in a wireless sensor network (WSN), is classification, that is, it is important to infer whether the samples measured by sensors in a WSN belong to a certain hypothesis (class) or not. It is well known that Support Vector Machines (SVMs) have been successfully used as classification tools in a variety of areas [1, 2, 3]. Training a SVM calls for solving a quadratic programming (QP) problem in a number of coefficients equal to the number of training examples. Because of this, for very large data sets, standard numeric techniques for QP become infeasible. To address the constraints associated with large data sets, various decomposition methods have been proposed. Some techniques decompose the problem into manageable subproblems over part of the data [4], while others perform iterative pairwise [5] or component-wise optimization [6].

A disadvantage of these techniques is that they may give only an approximate solution and may require many passes through the whole data set to reach a reasonable level of convergence. In principle, all working methods used to train SVMs, especially shrinking [4], can be considered as incremental learning algorithms, since only a small part of the samples is actually used for optimization in each step. However, all these approaches are not useful for true distributed learning in the context of WSNs, where there are important constraints in terms of memory and power available at the sensor nodes. This is because in all these methods, none of the samples are discarded during the training and thus all of them have to be considered in each working set selection step. As a consequence, both the memory and the power required are too high to be used in WSNs.

At the same time, as the research field of mobile computing and communication advances, so does the idea and the need of a distributed, ad-hoc wireless network of hundreds to thousands of microsensors, which can be randomly scattered in the area of interest. Network microsensors enable a variety of new applications such as environmental monitoring, warehouse inventory tracking, location sensing, patient and structural health monitoring. Moreover, in the near future, the development of visual sensor networking technology employing content-rich vision-based sensors will require efficient distributed processing for automated event detection and classification. Hence, the ability to incrementally learn from batches of data with minimal communication requirements is important for real-world applications. Distributed learning may be used to keep the memory and energy consumption of the learning algorithm at a manageable level as well as to make predictions at a time when the whole data is not yet available. This kind of incremental algorithms formulate the exact solution at step $i+1$ in terms of the solution at step i and the new set of available data samples.

An appealing feature of SVMs is the sparseness representation of the decision boundary they provide. The location of the separating hyperplane is specified via real-valued weights on the training samples. Training samples that lie far away from the hyperplane do not participate in its specification and therefore receive zero weight. Only training samples that lie close to the decision boundary between the two classes, the so-called *support vectors*, receive non-zero weights. Therefore SVMs seem well suited to be trained incrementally. In fact, since their design allows the number of support vectors to be small compared to the total number of training samples, they provide a compact representation of the data, to which new examples can be added as they become available. In our work, we take advantage of this compact representation in order to design an energy-efficient distributed learning algorithm for WSN.

Various incremental algorithms have been recently proposed [7, 8, 9, 10] for training a SVM. The key idea in all of them is to preserve only the current estimation of the decision boundary at each incremental step along with the next batch of data (or part of it). In this paper, we first provide in Section 2 a brief description of the key idea of SVMs; Section 3 presents two distributed algorithms for training a SVM as applied to the classification problem in a WSN. In Section 4, we present a set of simulation experiments in order to

This work was supported by the Greek General Secretariat for Research and Technology under Program ΠΕΝΕΔ, Code 03ΕΔ69.

assess the performance of our proposed approaches comparing them to the performance of a representative centralized SVM algorithm.

2. SUPPORT VECTOR MACHINES

Given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, support vector learning tries to find a hyperplane, determined by a vector $\mathbf{w}$ with minimal norm and an offset vector b, that separates the training data $\{\mathbf{x}_i\}$ into two classes denoted by $y_i = \{-1, +1\}$. Let $SVM = \{\mathbf{w}, b\}$ denote the separating hyperplane. To find such a hyperplane, one must solve the following quadratic problem [11]:

$$\min_{\mathbf{w},\xi} \Phi(\mathbf{w}, \xi) = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{n} \xi_i, \tag{1}$$

subject to

$$\begin{aligned} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) &\geq 1 - \xi_i \quad \text{and} \\ \xi_i &\geq 0, \qquad \text{for } i = 1, 2, ...n, \end{aligned} \tag{2}$$

where b determines the offset of the plane from the origin, the set of variables $\{\xi_i\}_{i=1}^n$ measures the amount of violation of the constraints, and C is a parameter that defines the cost of constraint violation. The vector of minimal norm $\mathbf{w}$ that represents the resulting separating hyperplane

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

is expressed by means of a linear combination of the so-called *support vectors*, i.e., the training sample vectors $\{\mathbf{x}_i\}_{i=1}^l$ corresponding to the l non-zero Lagrange multipliers $\{\alpha_i\}_{i=1}^l$, calculated during the optimization process [1]. In practical settings, the number of support vectors is usually quite small compared to the number of training samples ($l << n$). The decision function for classifying a new point $\mathbf{x}$ can be easily written as

$$f(\mathbf{x}) = sign\,(\sum_{i=1}^{l} y_i \alpha_i \mathbf{x} \cdot \mathbf{x}_i + b) \tag{3}$$

and the corresponding decision rule can be expressed as follows: a new test vector $\mathbf{x}$ belongs to class 1 when $f(\mathbf{x}) > 0$ while $\mathbf{x}$ belongs to class -1 when $f(\mathbf{x}) < 0$.

3. DISTRIBUTED TRAINING OF A SVM IN A WSN

Let us consider a deployment of m sensors taking measurements in a certain area. Our goal is to be able to train a SVM in an efficient and distributed fashion so that: a) we can get good classification results on test data and b) our algorithms can be used easily in the context of WSN, where the training must take place across sensors.

Notice that under the traditional centralized approach, the measurements should be sent first to a base station, where all the processing takes place and a decision boundary that separates the two classes is found. However, direct communication between each sensor and the base station (end-user) in a WSN is both cumbersome (due to the usually large number of sample vectors involved) and highly energy inefficient for a variety of reasons. First, the base station may be far away from the sensing area, and thus direct communication of raw sensor data to the base station can be quite energy costly. In addition, as the number of sensors in a network grows larger and larger, it becomes difficult to manage the vast amount of data collected from the sensors. Also, with increased node density in one location, multiple sensors may view the same event giving rise to sample vectors that are close to each other, and thus, may be redundant in terms of being useful to determining the separating plane.

On the other hand, as shown in previous work (e.g. [12]), in various practical problems related to WSN, it is possible to design energy-efficient clustering network protocols that greatly reduce the power dissipation. In such protocols, sensors are organized into local spatial clusters. Each cluster has a clusterhead, a sensor which receives data from all other sensors in the cluster, performs data fusion, and transmits the results to the base station. This greatly reduces the amount of data sent to the base station and thus achieves an improved energy efficiency. With this motivation, next, we propose two novel distributed algorithms in order to train incrementally a SVM in a WSN scenario using a energy-efficient clustering protocol.

3.1 Distributed Fixed-Partition SVM training

Typical fixed-partition techniques divide the training samples in batches clusters of sample vectors of fixed size [9]. These kind of algorithms seem appropriate for training incrementally a SVM using *only* partial information at each incremental step [8]. For the WSN scenario, we propose to use a Distributed Fixed-Partition algorithm (DFP-SVM) where the final estimation of the separating hyperplane is obtained incrementally through a sequence of incremental steps and where each incremental step takes place at a given cluster.

The key motivation behind this incremental algorithm is that as the number of support vectors is typically very small compared to the number of training samples, the data of previous clusters can be compressed to their corresponding estimated hyperplane (support vectors and offset). Thus, instead of transmitting to the next clusterhead all the measurements stored in the previous one, only the current estimation of the hyperplane is transmitted, which reduces very importantly the energy spent. More specifically, suppose there are K clusterheads in the sensor deployment. For each $i = 1, 2, \ldots, K$, the estimation $SVM_i = \{\mathbf{w}_i, b_i\}$ at clusterhead i is obtained combining the previous estimation SVM_{i-1} calculated at cluster $i-1$ and all the sample vectors measured by the sensors belonging to clusterhead i; after this estimation is obtained, the i-th clusterhead transmits SVM_i to the $(i+1)$-th clusterhead, Figure 1.

As we show in our experimental results of Section 4, after only a complete pass through all the clusters, a good approximation of the optimal separating plane is obtained, that is, the separating hyperplane is very similar to the one obtained using a centralized energy-inefficient algorithm, where all the sample data is used at once in a single training step at the base station.

3.2 Weighted DFP-SVM training

In many real world applications, the concept of interest (definition of classes to be separated) may be time-varying or space-varying; similarly, the underlying data distribution may change as well. Often these changes make the model built on old data inconsistent with the new data, hence regu-

[1] Notice that for simplicity, we assume that the sample vectors are enumerated such that the support vectors correspond (in any pre-agreed order) to the first l sample vectors

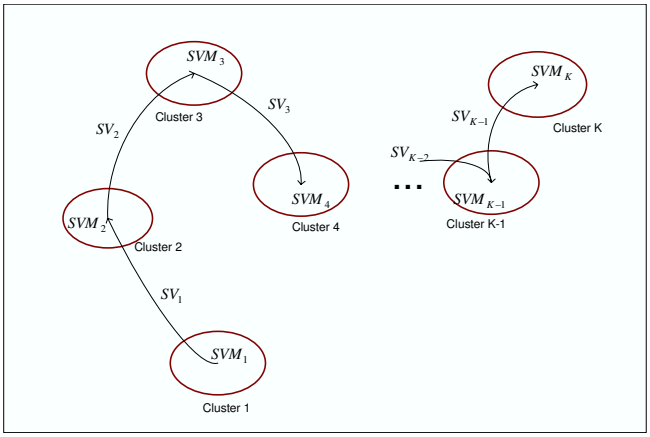

Fig. 1. Scheme of distributed training of a SVM: For each cluster, the estimation SVM_i at clusterhead i is obtained combining the support vectors (SV_{i-1}) of the previous estimation SVM_{i-1} calculated at cluster $i-1$ and all the sample vectors measured by the sensors belonging to cluster i.

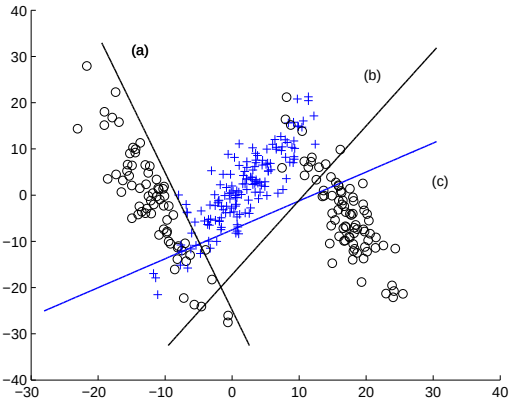

Fig. 2. Discriminant planes of the training set resulting from: (a) the first incremental step of the DFP-SVM, (b) the second incremental step of the DFP-SVM, and (c) the centralized algorithm.

lar updating of the model is necessary. This problem, known as *concept drift*, complicates the task of learning in SVM. A typical example of this phenomenon is weather prediction, where the rules may vary radically depending on the season.

On the other hand, one may also observe changes in the training data, which have no correspondence to controllable parameters of the experiment [13]. For example, in engineering applications, the quality of a machine deteriorates over the course of its life-cycle. Therefore, there is a need to have a robust system that can adapt easily to these uncontrollable changes.

In the case of distributed sequential training of a SVM in a WSN, this effect is even more accentuated: As the data is presented in several batches, changes in the target concept may occur between different batches of data. We are interested in possible concept drifts in WSN applications. For instance, consider a number of sensors distributed in a building, taking measurements of temperature, humidity and light in order to determine which rooms in the building are shinny or not. It is clear that the data distribution changes over time depending on the time of the day. Another example is vehicle tracking for surveillance or monitoring of a hostile environment. In this case, sensors should track all kinds of vehicles that pass through the area and probably have different characteristics such as weight, size, and shape.

We modify our previously proposed algorithm DFP-SVM in order to make it more suitable for WSN applications where there exist concept drifts. Our approach consists of adapting Ruping algorithm [7] to the WSN context. We call this algorithm as the Weighted Distributed Fixed-Partition SVM training (WDFP-SVM).

As an illustrative example, consider the 300 samples in Figure 2 taken by 300 sensors distributed in a field. Let us assume that the data is divided into two batches, such that the first cluster contains the sample vectors with $x < 0$. Training the SVM on this first cluster of data leads to the decision boundary denoted by line (a) in Figure 2. If we perform the distributed training for the SVM according to the DFP-SVM algorithm with two incremental steps, the resulting decision boundary (line (b)) largely ignores the old support vectors and it practically corresponds to the decision boundary that would have been learned if only the second cluster of samples had been used ignoring the first cluster. Although in general, this is a desired property of the SVM algorithm (because it means that the SVM is somehow robust against outliers), in the case illustrated in Figure 2, it can be seen that most of the outliers are the old support vectors, which causes an important misclassification error.

To address this problem, one needs to make the error on the old support vectors (representing the old learning set), more costly than the error on the new samples. This can be easily achieved by training the SVM with respect to a new loss function [7]. Let $(\mathbf{x}_i, y_i)_{i\in S}$ be the old support vectors and $(\mathbf{x}_i, y_i)_{i\in I}$ be the new sample vectors. The alternative cost function that should be used instead of (1) is:

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C(\sum_{i\in I} \xi_i + L \sum_{i\in S} \xi_i), \qquad (4)$$

where the parameter L increases the cost for the old support vectors. An appropriate heuristic choice for the parameter L is to let it be equal to the number of training samples in the previous cluster divided by the number of support vectors. This arises from the idea of approximating the average error of an arbitrary decision function (over all samples) by the average error calculated only over the support vectors. In this way, every support vector influences a constant fraction of all sample vectors.

4. RESULTS AND DISCUSSION

In this Section, we present a set of simulation experiments covering both the cases with and without concept drift in order to assess the performance of the two proposed distributed SVM algorithms. We evaluate our incremental algorithms comparing them to the traditional centralized SVM training algorithm [4]. At the same time, we also demonstrate that the energy consumption decreases when the SVM is trained incrementally as compared to the centralized case.

4.1 Performance of the DFP-SVM algorithm

In the centralized algorithm proposed in [4], there is only an evolving subset of sample data used, making it necessary to address all the constraints associated with large data sets. In

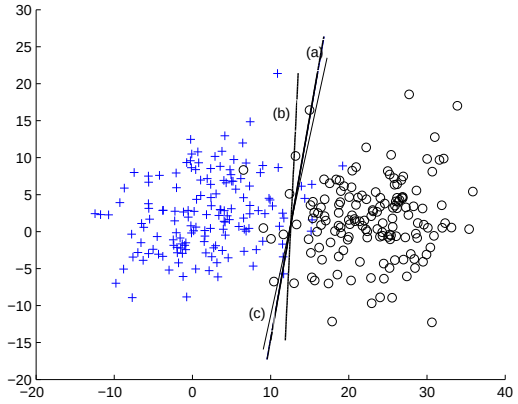

Fig. 3. Discriminant planes obtained using the centralized algorithm (line (a)) and the two proposed distributed algorithms DFP-SVM (line (b)) and WDFP-SVM (line (c)).

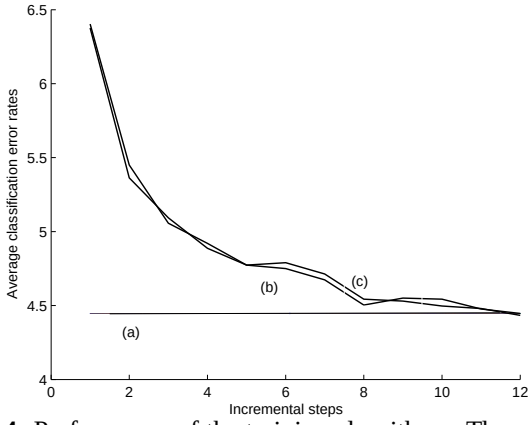

Fig. 4. Performance of the training algorithms: The average error rate of 500 Monte Carlo runs after training the SVM for consecutive incremental steps applying the centralized algorithm (line (a)), DFP-SVM (curve (b)) and WDFP-SVM (curve b(c)).

our WSN scenario, all the sample data is sent to the base station for processing so that none of the samples are discarded during the training and thus all samples are considered in each working set selection step.

We consider a sensor network composed of 300 nodes uniformly distributed in the field, where each of the sensors collects sample vectors from two classes. In our experiments, we generate the sample data of the two classes using two Gaussian distributions with two different mean values. Figure 3 illustrates the training set of 300 sample vectors generated by two Gaussian distributions with means $\vec{\mu}_1 = [2,2]$ and $\vec{\mu}_2 = [22,2]$ respectively. The corresponding representation ellipses are thus centered at $(2,2)$ and $(22,2)$, with eigenvalue ratios $\lambda_1 = \lambda_2 = \frac{35}{25}$ and rotation angles $\theta_1 = \theta_2 = 20^0$, respectively. The whole training set is partitioned into 12 clusters, each one with a fixed size of 25 sample vectors. Figure 3 illustrates our results. Plane (a) is the decision boundary found with the centralized algorithm. The planes denoted by (b) and (c), are the decision boundaries constructed after training the SVM using the DFP-SVM and WDFP-SVM algorithms, respectively. Both distributed algorithms give a good approximation of the decision boundary constructed with the centralized algorithm (plane (a)), in particular, the plane constructed using the WDFP-SVM algorithm (line (c)) coincides exactly with the one obtained using the centralized algorithm.

We also simulated 500 Monte Carlo runs in order to test the performance of these two distributed algorithms on another test data set drawn from the same distributions. Figure 4 represents the average error rates (%) for our two proposed algorithms as a function of the consecutive incremental steps. At each step, only the hyperplane parameters are used together with the sample vectors of the next cluster of nodes, and it is shown that with only one pass across the clusters, both distributed algorithms converge to the same average error rate obtained with the centralized algorithm, which requires more energy.

On the other hand, Figure 5 simulates a scenario with a concept drift. The first cluster of data consists of measurements of two Gaussians with mean vectors $\vec{\mu}_1 = [2,2]$, $\vec{\mu}_2 = [-12,-2]$, eigenvalue ratios $\lambda_1 = \frac{2}{1}$, $\lambda_2 = \frac{35}{25}$ and rotation angles $\theta_1 = 90^0$, $\theta_2 = 20^0$, respectively. The decision boundaries for this subset of training vectors that is obtained

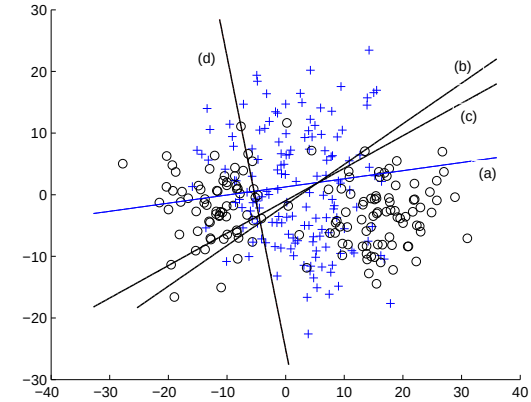

Fig. 5. Concept drift: Discriminant planes obtained with the centralized algorithm (line (a)), the DFP-SVM algorithm (line (b)) and two consecutive steps of the WDFP-SVM (lines (c),(d)).

using DFP-SVM and WDFP-SVM, are coincident (line (d)). At the next step, once the parameters of the estimated hyperplane are transmitted to the next cluster, in order to introduce the concept drift, we now assume that the next batch of sample vectors consists of samples from the following 2 classes: one class is the same first Gaussian of the previous batch (mean vector $\vec{\mu}_1 = [2,2]$), while the other class consists of a shifted Gaussian with mean vector $\vec{\mu}_2 = [16,-3]$. Since the DFP-SVM algorithm ignores the hyperplane obtained from the fist batch of sample vectors, the resulting plane illustrated in Figure 5 (line (b)) almost corresponds to the decision boundary that would have been learned using only the second batch of samples alone. However, the WDFP-SVM constructs a plane (line (c)) that lies much closer to the result obtained in the centralized case (line (a)).

4.2 Energy efficiency of the DFP-SVM algorithm

At this point we would like to investigate the benefits in terms of energy in a wireless sensor network using these distributed algorithms for training a SVM. Specifically, we are interested in the comparison of energy consumed by the proposed

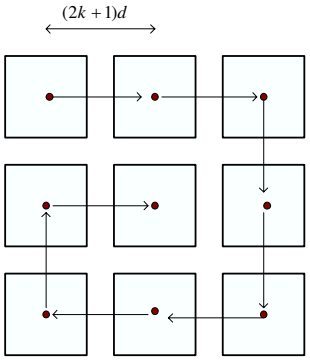

Fig. 6. Transmission path: Each clusterhead (black dot) transmits the support vectors to the next clusterhead.

distributed algorithm to a scheme where all sensors transmit their data to a fusion center for processing.

The total energy consumed in the distributed training can be expressed as the sum of the energy consumed in each cluster and the energy consumed for the transmission of the support vectors to the next clusterhead. The energy cost for the transmission of a measurement from node A to node B is proportional to the squared distance of node A to B.

Consider the arrangement of n sensors in a cubic lattice where each sensor is at distance d of a neighbor sensor. Now, separate the sensors in K clusters of $(2k+1)\times(2k+1)$ sensors each. Each sensor consumes $E_K(d)$ energy for transmitting its measurements to the clusterhead and each cluster consumes $E_{sv}(d)$ energy for the transmission of N_i support vectors to the next clusterhead $i+1$.

The total energy consumed for the distributed training of a SVM after one pass of all K clusterheads through the path depicted in Figure 6 using the proposed algorithms is $E_d(d) = E_{sv}(d)+E_K(d)K$, or:

$$E_d(d) = (2k+1)d^2(N_1+N_2+\ldots+N_{K-1}) + (6d^2k(k+1)+8d^2\sum_{j=1}^{k-1}\sum_{i=1}^{k-j}2(k-i)+\sum_{j=1}^{k-1}j(k-j))K.$$

On the other hand, the energy cost for the direct transmission of the measurements of $(2k+1)\times(2k+1)$ sensors to the base station is given by the expression:

$$E_c(d) = 8d^2\sum_{j=1}^{k-1}\sum_{i=1}^{k-j}(i^2+(k-i+1)^2)+2d^2k(k+1)(2k+1).$$

We simulated 500 Monte Carlo runs in order to estimate the energy consumed during the distributed training of a SVM. For a scenario of $n = 225$ sensors in a square grid arrangement separated in $K = 9$ clusters consisting of 25 sensors each (hence $k = 2$), the energy cost for the training of the SVM using the proposed distributed algorithm is $E_d(d) = 3380d^2 + 9\cdot 60d^2 = 3920d^2$, while in the centralized case the cost is $E_c(d) = 8400d^2$. This simulation experiment shows that the proposed distributed algorithm is much more efficient in terms of energy consumption than the centralized algorithm, since it reduces the energy cost by more than 50%.

5. CONCLUSIONS

We introduced the concept of distributed training of a SVM in a wireless sensor network. Our research was motivated by the need to have energy-efficient distributed algorithms to be used in large-scale WSNs, whose goal is to perform classification tasks. We presented two distributed algorithms for training a SVM in a WSN. The DFP-SVM algorithm constructs a hyperplane that converges to the plane, which is very close to the one obtained with a centralized algorithm. On the other hand, for the case where concept drift is present, we proposed the WDFP-SVM algorithm which adapts to the non-stationarity. We presented several simulation experiments in order to assess the performance of our proposed approaches. Both algorithms performed only one sequential pass through clusters of sensors.

REFERENCES

[1] T. Joachims, "Text categorization with support vector machines," in *Proc. 10th Eur. Conf. on Machine Learning (ECML'98)*, Chemnitz, Germany, April 1998, pp. 137–142.

[2] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in *Proc. of the 1997 Conference on Comp. Vision and Pattern Recogn.*, Washington, DC, USA, 1997, p. 130, IEEE Computer Society.

[3] A. Bulut, P. Shin, and L. Yan, "Real-time nondestructive structural health monitoring using support vector machines and wavelets," in *Proc. of the Conf. on Advanced Sensor Technologies for Nondestructive Evaluation and Structural Health Monitoring (NDE'05)*, San Diego, CA, USA, March 2005.

[4] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE Workshop on Neural Networks and Signal Processing, NNSP'97*, Amelia Island, FL, September 1997, pp. 276–285.

[5] J. Platt, *Fast Training of Support Vector Machines using Sequential Minimal Optimization*, MIT Press, 1999.

[6] T. T. Friess, N. Cristianini, and C. Campbell, "The kernel adatron algorithm: a fast and simple learning procedure for support vector machines," in *Proc. 15th Int. Conf. on Machine Learning*, Madison, Wisconsin, July 1998, pp. 188–196.

[7] S. Ruping, "Incremental learning with support vector machines," in *Proc. IEEE Int. Conf. on Data Mining*, San Jose, CA, USA, November 2001, pp. 641–642.

[8] N. Syed, H. Liu, and K. Sung, "Incremental learning with support vector machines," in *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence, IJCAI'99*, July 1999.

[9] C. Domeniconi and D. Gunopoulos, "Incremental support vector machine construction," in *IEEE Int. Conf. on Data Mining, ICDM'01*, San Jose, CA, USA, November 2001.

[10] C. P. Diehl and G. Cauwenberghs, "Support vector machine incremental learning, adaptation and optimization," in *Proc. Int. Joint Conf. on Neural Networks*, Portland, OR, July 2003.

[11] K. Bennett and C. Campbell, "Support vector machines: hype or hallelujah," *SIGKDD Explorations*, vol. 2,2, pp. 1–13, 2000.

[12] A. Wang and A. Chandrakasan, "Energy efficient DSPs for wireless sensor networks," *IEEE Signal Proc. Magazine*, 2002.

[13] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proc. Int. Conf. on Machine Learning*, Stanford, CA, USA, 2000, pp. 487–494.