TOWARDS ANALOG IMPLEMENTATION OF SUPPORT VECTOR MACHINES: A TIME-CONTINUOUS FORMULATION OF THE CLASSIFICATION PHASE

G. OLTEAN, M. GORDAN
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, ROMANIA

KEYWORDS: Support vector machine, Fast classification, Time-continuous signal processing, Analog implementation

ABSTRACT: Support vector machines (SVMs) are powerful classifiers for large tasks as image/video-sequence analysis. A practical implementation issue for such problems is their real time operation in the classification phase. One recent strategy is the development of algorithmic formulation of SVMs suitable for hardware implementation. We propose such an approach, that allows a partially sequential – partially parallel implementation of the SVM classification, through the description of the feature vectors as time-continuous signals. This allows a simple analog implementation of the dot product between each test and support vector in a sequential fashion and a parallel computation of all the dot products for a test vector. The functionality of the algorithm is validated through a Simulink model and a set of experiments on the IRIS dataset. The solution will offer a good speed-complexity compromise in SVM classification implementation.

INTRODUCTION

Support vector machines (SVMs) represent a powerful machine learning technique that gained recently a great interest from the scientific community, especially due to their performance in solving difficult classification and pattern recognition tasks. One of the most noticeable application fields where SVM classifiers proved their very good performance is image analysis. Several applications are recently reported in face detection, object tracking in video-sequences, face recognition, facial feature localization etc. [1,2]. These are difficult recognition tasks, due to the variability of the appearance of the same pattern and to the difficulty of defining and extracting reliable features to describe the patterns, so that to maximize the inter-class and minimize the intra-class variance. SVM classifiers prove able to learn from small sets of (often sparse) examples, without the need of a carefully selected feature extraction strategy, with very small recognition error and high generalization performance.

In its basic form, an SVM is a binary classifier based on the optimal separating hyperplane algorithm that implements the Structural Risk Minimisation principle. In its training phase, the SVM receives at its input a set of labelled training patterns in the form $\{\mathbf{x}_i, y_i\}$, $i=1,2,...,N_{trn}$, where $\mathbf{x}_i$ is a vector of N real-valued features, $\mathbf{x}_i \in \Re^N$, and y_i is its label, $y_i \in \{-1;+1\}$; a value +1 is assigned to a positive example for the classifier, whereas –1 is assigned to a negative example. Based on the training set the SVM learning algorithm derives the so-called optimal separating hyperplane. This will (ideally) perfectly separate the positive from the negative examples, ensuring a maximal distance between the closest positive and the closest negative example to the hyperplane [3]. The training patterns $\mathbf{x}_s$, $s=1,2,...,N_s$, $N_s<<N_{trn}$, that are the closest positive and closest negative to the hyperplane are called *the support vectors* of the SVM classifier. The support vectors, together with their associated positive Lagrange multipliers α_s and with their labels y_s, completely define the decision function of the SVM classifier, in the form:

$$f(\mathbf{x}) = \sum_{s=1}^{N_S} \alpha_s y_s K(\mathbf{x}, \mathbf{x}_S) + b, \qquad (1)$$

where $\mathbf{x}$ denotes an unlabeled pattern to be classified by the trained SVM, $\mathbf{x} \in \Re^N$; b represents the bias term of the hyperplane; $K(\cdot,\cdot)$ represents a kernel function used to compute the dot product between $\mathbf{x}$ and $\mathbf{x}_s$, either in their original space $\Re^N$ (for a linear SVM) or in a higher dimensional feature space $\Re^M$, M>>N, where the data are projected to become linearly separable (for non-linear SVMs) [3]. Typical forms of kernels are:

a) For the linear SVM:

$$K(\mathbf{x}, \mathbf{x}_S) = \mathbf{x} \cdot \mathbf{x}_S = \mathbf{x}^T \mathbf{x}_S \qquad (2)$$

b) For a non-linear SVM: the polynomial kernel of degree d, with $p \in \Re, q \in \Re$ - coefficients:

$$K(\mathbf{x}, \mathbf{x}_S) = \left(p \cdot \mathbf{x}^T \mathbf{x}_S + q\right)^d \qquad (3)$$

The number of the support vectors N_s increases with the complexity of the classification task. Furthermore, difficult classification problems are in general associated to a large dimensionality of the pattern space, i.e. N – large. Therefore the more difficult the classification problem, the larger the computational complexity of the SVM classification process, both in the training phase [4] and in the classification phase [5]. Although a great effort was devoted to find efficient computational implementations both for the training phase (in which the tractability of the optimisation process leading to the SVM classifier is the most important issue) and for the test phase (where the problem is the real-time evaluation of $f(\mathbf{x})$), in this paper we address only the latter aspect of

the problem. Reducing the numerical complexity of the classification phase is extremely important for applications that must run in real-time, as e.g. video-sequence analysis [2], therefore this topic is of actual interest for the scientific community.

The computational complexity of the classification phase can be evaluated in terms of the number of elementary operations needed for the evaluation of f(**x**) in every unlabeled pattern **x**. Examining the equations (1), (2) and (3), one can see that each evaluation requires N_s dot product computations in the form $\mathbf{x}^T\mathbf{x}_s$, s=1,2,...,N_s, followed by N_s kernel evaluations for every $\mathbf{x}^T\mathbf{x}_s$, N_s multiplies with the constant $y_s\alpha_s$ and N_s-1 additions. The dot product is expressed as:

$$\mathbf{x}^T\mathbf{x}_s = \sum_{i=1}^{N} x_i x_{si} \ , \qquad (4)$$

$$\mathbf{x} = [x_1 \quad x_2 \quad ... \quad x_N]^T ; \mathbf{x}_s = [x_{s1} \quad x_{s2} \quad ... \quad x_{sN}]^T .$$

Thus to evaluate every $\mathbf{x}^T\mathbf{x}_s$ one needs N multiplies and N-1 additions, which leads to a total of approximately $2N \cdot N_s$ operations per pattern classification. Furthermore, considering that many image analysis tasks decompose the image into a set of P patterns (partially overlapping image windows), the number of operations needed for the complete classification increases P times, to $2P \cdot N \cdot N_s$ [1,2]. Therefore, speeding-up the classification phase becomes a non-trivial task for real-world applications of SVM classifiers. The types of solutions reported in the literature depend of the type of implementation:

1) Solutions devoted to software implementations of SVM classifiers. In this case, the computational time can be decreased only by reducing the number of support vectors N_s or the dimension of the feature space N since all the computations are performed sequentially. N_s can be minimised by selecting only the most significant support vectors [5]. To reduce the feature space dimension N, various data compression techniques can be used as e.g. principal component analysis [6].

2) Solutions achieved by the hardware implementation of SVM classifiers. In this case, the duration of the classification phase is reduced by parallelising to the largest possible extent the computation of f(**x**). Analog implementations are recently reported in the literature [7]. It is easy to see from equation (1) that all the evaluations of the kernel function $K(\mathbf{x},\mathbf{x}_s)$, s=1,2,...,$N_s$, and the N feature-by-feature multiplies from the computation of $\mathbf{x}^T\mathbf{x}_s$ can be performed in parallel. This leads to massively parallel circuit structures.

However, as the parallelism increases, the circuit level complexity of the SVM classifier increases as well. A massively increased complexity resulting for a large-scale classification problem might be unpractical for certain applications. A compromise solution would be a partially sequential – partially parallel operation, lying somewhere between the software and the hardware implementation; this could provide a good trade-off between the duration of the classification phase and the hardware complexity, thus being also more suitable for large-scale problems. Such a solution is proposed in this paper. The proposed approach is the implementation of each dot product $\mathbf{x}^T\mathbf{x}_s$, s=1,2,...,N_s, in a sequential analog fashion, whereas the N_s dot product computations (and of course kernel evaluations) will be done in parallel. A number of N_s parallel analog circuit structures will be needed for the evaluation of f(**x**). The computation of $\mathbf{x}^T\mathbf{x}_s$ in the analog sequential way is achieved by providing time-continuous descriptions of the pattern **x** and support vectors $\mathbf{x}_s$, using signal reconstruction techniques from their samples. With the time-continuous formulation of **x** and $\mathbf{x}_s$ as analog signals x(t) and $x_s(t)$, the computation of $\mathbf{x}^T\mathbf{x}_s$ can be formulated as an analog signal multiplication followed by a signal integration; these operations can be implemented with simple analog structures.

In this paper we prove, through a mathematical demonstration, a Simulink model implementation and through a set of experimental results performed with the implemented Simulink model on a standard set for data classification, the validity of the proposed algorithm. Its circuit level implementation is currently an issue of our ongoing research work in the field.

TIME-CONTINUOUS FORMULATION OF THE CLASSIFICATION PHASE

The motivation for the proposed time-continuous formulation of the SVM classification phase originates from a typical application of SVM classifiers: the classification of grey-level images based on their content [1]. In the simplest implementation, the feature vector for such a task is obtained by scanning the grey-level digital image in row order. Thus every pattern **x** is just an ordered collection of grey levels, as illustrated in Figure 1a). However, one could consider not the digital image representation, but the analog image representation as in classical television applications. The analog version of the image can be represented as a sequence of time-continuos signals corresponding to each scan line as in Figure 1b). The digital image is the sampled and quantized version of the analog signal. Thus **x** can also be represented as the sampled and quantized version of the analog image, as in Figure 1c).

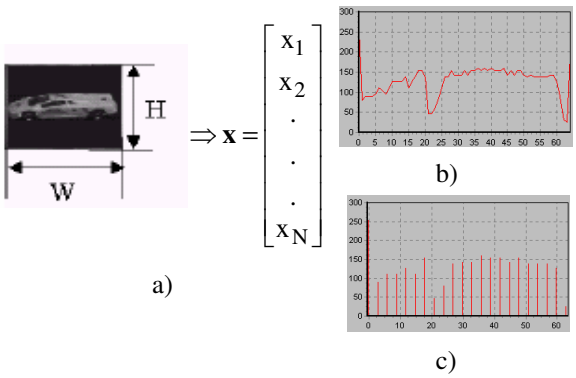

Fig. 1. Image description: a) as a digital image and in vector form; b) in analog form (one scan line); c) alternate representation as a sampled and quantized signal

Let us denote the description of the pattern **x** as a sampled signal by X(t).

In analytical form, the sampled signal X(t) can be expressed as follows:

$$X(t)=\begin{cases} x_i, \text{if } t=t_i=\dfrac{i}{f_S}; i=0,1,...,N-1 \\ 0, \text{otherwise} \end{cases} \quad (5)$$

In equation (5), by f_S we denote the sampling frequency of the original analog signal from Figure 1b) to get the sampled signal in Figure 1c). Of course every support vector $\mathbf{x}_s$ can be expressed just in the same fashion by a corresponding sampled signal $X_s(t)$, s=1,2,...,N_s.

We can consider that the (for this case, available) analog versions of the images representing the pattern $\mathbf{x}$ and the support vectors $\mathbf{x}_s$, namely the time-continuous signals denoted by x(t) and $x_s(t)$, are just more complete versions of the sampled signals X(t) and $X_s(t)$. Then if we can express the decision function f($\mathbf{x}$) of the SVM classifier given by the equation (1) in respect to x(t) and $x_s(t)$, s=1,2,...,N_s, we will obtain a time-continuous formulation of the SVM classification phase. For the time being, we will consider in equation (1) the use of the linear kernel or polynomial kernel, as in these functions $\mathbf{x}$ and $\mathbf{x}_s$ appear only in the form of their dot product, $\mathbf{x}^T\mathbf{x}_s$. Actually this is the only term that needs to be expressed in a time-continuous fashion, according to our implementation goal formulated in the previous section.

Describing $\mathbf{x}$ and $\mathbf{x}_s$ as the sampled signals X(t) and $X_s(t)$, $\mathbf{x}^T\mathbf{x}_s$ given by equation (4) can be expressed as:

$$\mathbf{x}^T\mathbf{x}_S=\sum_{i=1}^{N}X(t_i)X_S(t_i), \quad (6)$$

which furthermore, neglecting the effect of quantization, can be expressed in terms of the analog time-continuous signals x(t) and $x_s(t)$ as:

$$\mathbf{x}^T\mathbf{x}_S=\sum_{i=1}^{N}x(t_i)x_S(t_i). \quad (7)$$

Equation (7) can be written in continuous form as:

$$\mathbf{x}^T\mathbf{x}_S=\int_{t=0}^{N/f_S}x(t)x_S(t)dt, \quad (8)$$

which gives the basic of the time-continuous formulation of the SVM classification proposed.

Although this time-continuous formulation was initiated considering an image classification application, it can be extended to any types of patterns $\mathbf{x}$, regardless if their features are of the same kind, as long as they are real valued. However in the latter case, no time-continuous analog version of $\mathbf{x}$ and $\mathbf{x}_s$, s=1,2,...,N_s, was ever available. Therefore signal reconstruction methods as e.g. generation of quantized but not sampled signals or interpolation techniques must be applied to get analog versions of the patterns. In practice, depending on the reconstruction method, one might get as the result of the integration a larger value than $\mathbf{x}^T\mathbf{x}_s$. However as long as the result is proportional to the dot product by a known constant factor, this is not a problem for the computation.

With this formulation, the SVM classification phase can be implemented using the equations (8), (1) and (depending on the type of SVM) (2) or (3), as follows:

1) A sequential computation of each dot product $\mathbf{x}^T\mathbf{x}_s$, s=1,2,...,N_s using an analog multiplier block that has at its inputs the signals x(t) and $x_s(t)$, followed by an analog integrator whose output is read at the moment N/f_S after the integration start and multiplied by the constant $\alpha_s y_s$. This is illustrated in Figure 2 which is the detailed description of the blocks "Weighted sum" from Figure 3.

2) Parallel computations of all the dot products $\mathbf{x}^T\mathbf{x}_s$ for the N_s support vectors. This process requires a number of N_s structures as the one given in Figure 2. For a linear SVM, the N_s scalar outputs simply enter a summation block along with the bias term b. For a non-linear SVM with polynomial kernel, every scalar output $\mathbf{x}^T\mathbf{x}_s$ is first processed according to equation (3) prior to summation.

Usually the number of support vectors N_s is much smaller than the length of the feature vector N (in practical applications – at least by a factor of 10). Therefore the circuit complexity (defined as the number of simple analog building blocks needed for the implementation) is much smaller than in the massively parallel structures, which makes the approach suitable even for large scale SVM classification problems. On the other hand the computational speed, although lower than in the case of a fully parallel system, is higher than in the case of a fully sequential system.

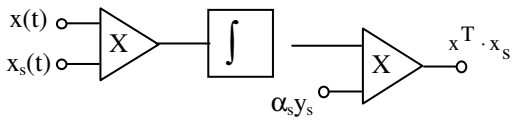

Fig. 2. Block diagram illustrating the sequential analog implementation of the dot product $\mathbf{x}^T\mathbf{x}_s$

The last issue to complete the proposed formulation of the SVM classification phase is the generation of the time-continuous signals x(t) and $x_s(t)$ through signal reconstruction techniques, from their sampled versions X(t) and $X_s(t)$, s=1,2,...,N_s.

Two issues should be mentioned here:

1) Assuming the same sampling frequency f_S for X(t) and for any $X_s(t)$, any pair of sampled signals in the form (X(t), $X_s(t)$) will ideally be perfectly synchronised. However in practice this case will never hold. Thus if one wants to use directly the sampled signals as inputs in the computational block given in Figure 2, if any time delay between X(t) and $X_s(t)$ appears, a large error will appear in the resulting dot product.

2) In the general case when $\mathbf{x}$ and $\mathbf{x}_s$ do not originate from analog, time-continuous signals, a value $f_S \to \infty$ (corresponding to the ideal sampling) will not be as beneficial as for analog signals. On the contrary, the time-continuous signal multiplication will be again prone to errors, as in the first case considered here, for any time delay between the 2 signals.

In order to avoid these problems, one needs to provide suitable reconstructed waveforms x(t) and $x_s(t)$ so that, even in the presence of a time delay τ_d, the dot product to be computed with a very small error. Two simple solutions to produce signals x(t) and $x_s(t)$ that satisfy this condition are given in the following:

1) **The most simple solution** is to consider x(t) and $x_s(t)$ in the form of some quantized but not sampled signals. In this case, "the restoration" of the time-

continuous signals is achieved by repeating, on each time interval $[t_k;t_{k+1})$, $t_k=k/f_S$, k=0,1,...,N-1, the corresponding sample (feature) value, namely, x_{k+1} for x(t) and $x_{s,k+1}$ for $x_s(t)$. Thus x(t) and $x_s(t)$ will have the following expressions:

$$x(t) = x_{k+1} \text{ for } t \in [t_k ; t_{k+1}), \forall k = 0,1,..., N-1 \quad (9a)$$

$$x_s(t) = x_{s,k+1} \text{ for } t \in [t_k ; t_{k+1}), \forall k = 0,1,..., N-1 \quad (9b)$$

The resulting signals x(t) and $x_s(t)$ for some $s \in \{1,2,..., N_s\}$ are illustrated in Figure 5 of the Experimental results section, for a particular example with N_s=2 and N=4 (the plots denoted XTest and SV1 on Scope 1). They can be considered periodical square wave signals, with the period N/f_S.

Thus, in the ideal case of no time delay between x(t) and $x_s(t)$, at the output of the integrator from Figure 2 we will have at the moment N/f_S after the start of the integration, the numerical value:

$$\int_0^{N/f_S} x(t)x_s(t)dt = \sum_{k=0}^{N-1} \int_{k/f_S}^{(k+1)/f_S} x(t)x_s(t)dt = \frac{1}{f_S}\sum_{i=1}^{N} x_i x_{si} = \frac{1}{f_S}\cdot \mathbf{x}^T\mathbf{x}_s, \; s = 1,2,..., N_s, \quad (10)$$

that is proportional to the dot product by a factor of $1/f_S$. If f_S=1, equation (10) gives exactly the dot product $\mathbf{x}^T\mathbf{x}_s$.

2) Another simple solution would be to generate the time-continuous signals x(t) and $x_s(t)$, s=1,2,...,N_s, from their discrete samples by interpolation. The most simple procedure is the linear interpolation. The advantage of such an approach in the generation of x(t) and $x_s(t)$ is that the signals will not have any local discontinuities. However when x(t) and $x_s(t)$ are piecewise linear, their product on each time interval $[t_k;t_{k+1})$ will be a 2nd degree polynomial, thus its integral will no longer be proportional to the dot product $\mathbf{x}^T\mathbf{x}_s$ by a constant factor. Therefore better interpolation methods should be found to generate x(t) and $x_s(t)$. The investigation of such methods will make the object of our future work.

A SIMULINK MODEL OF THE TIME-CONTINUOUS SVM CLASSIFIER

The model of the time-continuous implementation of the classification phase using SVM is shown in Figure 3. This is a general bloc diagram that can be used for any number of support vectors N_s. In this model, we describe each support vector by the pair $(\mathbf{x}_s,y_s)$. The support vectors, their corresponding Lagrange multipliers α_s, s=1,2,...,N_s, and the bias term of the hyperplane b are previously detected in the training phase of the support vector machine. Our implementation of the classification phase is independent of the training method (i.e. software or hardware).

The signals entering the classification phase are:

- The support vectors, denoted as Support Vector s, s=1,…N_s, each vector being described by its corresponding time-continuous signal $x_s(t)$ and its label y_s (as a time constant).
- The Lagrange multipliers α_s, denoted by alpha s, s=1,2,...,N_s, as time constants.
- The pattern to be classified **x**, denoted by Xtest, described in the time-continuous fashion as x(t) given in the previous section.
- The bias term b, denoted as Bias, as a time constant.

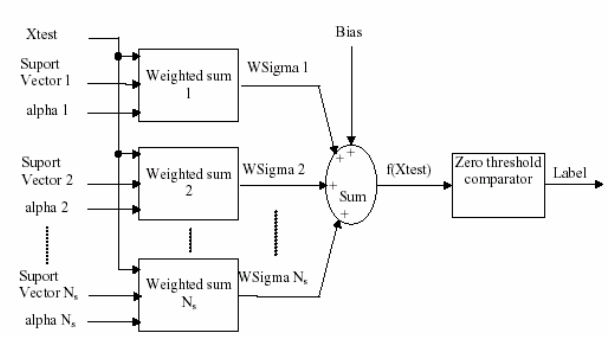

Fig. 3. Block diagram of the time-continuous SVM classifier

The number of the computational blocks "Weighted sum s" equals the number of support vectors, to compute the weighted sum of the corresponding support vector and Xtest vector. The resulting time continuous signal from these blocks, WSigma s, are added together with the bias term, Bias, by the Sum block. The result is the signal f(Xtest), that gives the evolution of the decision function of the classifier evaluated in Xtest as a time-continuous signal. It worth to mention that the intermediate values of f(Xtest) show the effect of each individual feature on the decision function. This is a specific characteristic of our implementation as compared to other implementations. Anyway the final value of the decision function will be read as the value of f(Xtest) at the moment N/f_S, as described in the previous section. The final classification result, i.e. the Label, is provided by the Zero threshold comparator block, as +1 if f(Xtest)>0, and -1 if f(Xtest)<0.

To verify the operation of the designed analog SVM classifier, we implemented it in Simulink, under Matlab environment. To keep the implementation simple, in order to observe the full behavior of the classifier, we chose a classifier with two support vectors. The detailed Simulink model is presented in Figure 4. As one can see the Simulink model has two computation channels, the first one corresponds to the first support vector (upper part) and the second one corresponds to the second support vector (lower part).

First of all we generate the continuous time signals corresponding to the features of all involved vectors:

- Xtest signal for test vector, using the block TestVect;
- SV1 signal for first support vector, using the block SupVect1;
- SV2 signal for second support vector, using the block SupVect2.

An example of these time-continuous signals can be seen in the experimental results section, in Figure 5.

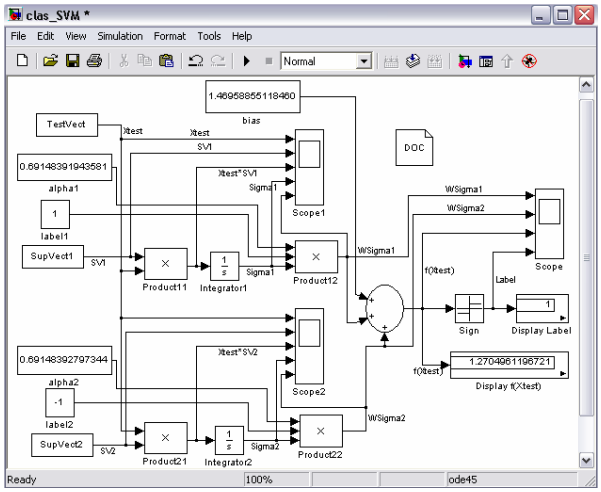

Fig. 4. The Simulink model of the time-continuous SVM classifier

These signals are collected with Scope1, Scope2 and Scope in the Simulink model. In our model we considered a duration of 1s for each feature of the input vector. The Product11 and Product21 blocks are responsible for the multiplication of the test vector (the signal Xtest) with the first support vector (the signal SV1) and with the second support vector (the signal SV2), thus producing the time-continuous signals Xtest*SV1, Xtest*SV1. The integration of each product signal by the blocks Integrator1 and Integrator2 gives the cumulated sum corresponding to the dot product. The resulting signals are Sigma1 and Sigma2, that are further multiplied by the corresponding Lagrange multiplier and the support vector's label. The resulting weighted signals are WSigma1 and WSigma2. Finally, from these "partial" signals, the decision function f(Xtest) is obtained according to the equation (1) applied for a linear SVM, using a summation block with three inputs: WSigma1, WSigma2 and Bias.

The signal f(Xtest) on the Scope represents the evolution in time of the decision function of the classifier. Although the classification result is given by the value of f(Xtest) only at the moment N/f_S (in our case, t=4s), one can notice that the proposed implementation allows us for an even better observation of the SVM classification process, as follows. We can consider in our example four significant time moment: 3 intermediate and one final. At t=1s, the decision function contains only the classification result according to the first feature and so on, up to t=4s, when we get the final value of the decision function we are interested in. Thus, in order to reduce the computation time, with a proper ordering of the features according to their significance, it yields possible to consider as final value of the decision function the value at an intermediate time moment, after taking into consideration only the most significant features.

The label assignment to the unknown pattern Xtest to be classified is implemented by the Sign block, in the Simulink model. This block is a simple comparator having the zero value for its threshold. The value of the output signal of this block, Label, should be read et the moment t=4s, after all the vectors' features were processed.

The Simulink model can be very easy extended for any number of support vectors by simply replicating the computing channels and adding the necessary inputs to the summation block.

Our implementation is also very useful from the didactical point of view. Due to the fact we have access and can see the signals in all the intermediate points of the computing flow, one can easy understand the operations involved in the classification phase of the SVM. Also, the verification of our future implementation ideas mentioned in the previous section is very easy with this model, due to its interactive mode of operation.

EXPERIMENTAL RESULTS

We tested our implementation on a standard data set for classification tasks, namely, the IRIS data [8]. Each pattern in the data set is described by 4 features of an iris flower: the petal length and width and the sepal length and width. There are 3 classes of irises: Setosa, Versicolor and Virginica. The goal is to classify each individual pattern in one of the 3 classes. We consider in our experiment the binary classification of an unknown pattern in the class Setosa vs. the other 2 classes. For the SVM training (needed to obtain the support vectors, their Lagrange multipliers and the bias term) we used Steve Gunn's Matlab application [9]. The IRIS data set contains 75 training patterns. The training is performed for a linear SVM with the error penalty parameter C=1. After the training phase we get 2 support vectors, whose features, labels and Lagrange multipliers are presented in Figure 4 and 5, along with the bias term of the resulting classifier.

As test vectors, we selected 12 patterns from the standard IRIS test set to be classified using two implementation: Steve Gunn's implementation (considered here as reference classifier, and denoted as SG) and our Simulink SVM classification implementation (denoted as SCT). The values of the decision function in the 2 implementations along with the "target" classification results are presented in Table 1. Since the classification labels always match the target, we do not list them in the table, but only the target ones, denoted there as simply "Label".

As one can notice by examining the results in Table 1, the differences in the real value of the decision function (error column in Table 1) computed by SG implementation and our SCT implementation can be considered zero, since their order is 10e-7 for all the test vectors. So our SCT implementation always provides correct values, for various signs and magnitudes of the decision function.

The computational details in each step of the algorithm can be seen in Figure 5. This figure illustrates the waveforms for all the input, intermediate and output signals during the classification of the test vector 1 from Table 1. The signal Xtest is very similar with the signal SV1 but different from the signal SV2. This observation is very important, being a qualitative information about

the membership of the test vector to the same class as support vector 1. The time variation of the decision function f(Xtest) is very important because it offers information about an intermediate membership of the test vector after taking into consideration only the firsts features of the test vector, after each time period. In accordance with the first feature (time moment t =1s), f(Xtest)=0.364>0, label=+1, so the test vector belongs to the Setosa class. Also considering the first two featurs (t=2s), three features (t=3s), or all four features (t=4s) we have f(Xtest)>0, label=+1, so Setosa class.

TABLE 1. Classification results

Xtest	f(Xtest)			Label
	SG	SCT	error	
1.	0.8415397971	0.8415394766	-3.20e-07	1
2.	0.9282835252	0.9282832184	-3.06e-07	1
3.	0.7703558444	0.7703555818	-2.62e-07	1
4.	0.9946513779	0.9946510561	-3.22e-07	1
5.	1.2129277395	1.2129273884	-3.51e-07	1
6.	1.7282735109	1.7282731483	-3.63e-07	1
7.	-4.5566642408	-4.5566646840	-4.43e-07	-1
8.	-1.2689215012	-1.2689217966	-2.95e-07	-1
9.	-2.3058037663	-2.3058041356	-3.69e-07	-1
10.	-2.2193649549	-2.2193653424	-3.87e-07	-1
11.	-1.3893638414	-1.3893641758	-3.34e-07	-1
12.	-3.3223390851	-3.3223394769	-3.91e-07	-1

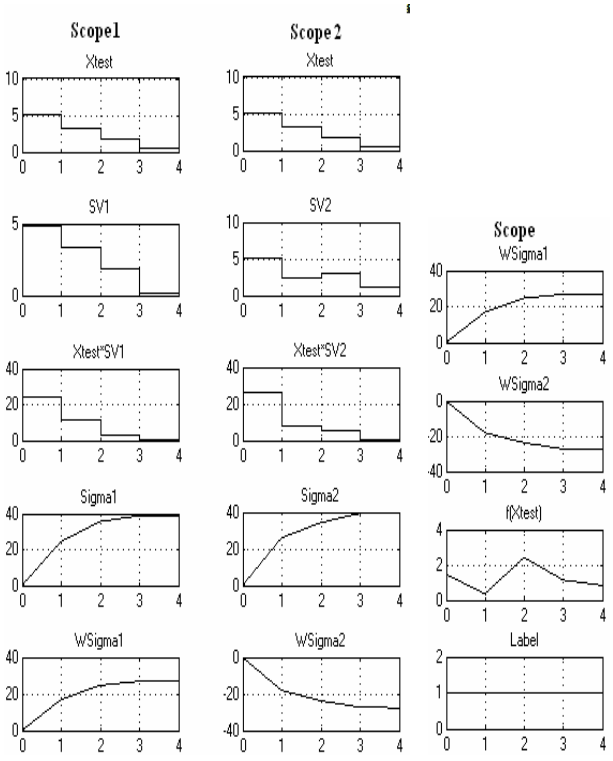

Fig. 5. Classification of the test vector 1 using the SCT implementation

CONCLUSIONS

In this paper we proposed a new algorithm for the implementation of the classification phase in SVM classifiers. The aim is a good compromise solution between the duration of the classification phase and its complexity in analog hardware implementation. The proposed algorithm is based on the description of the feature vector $\mathbf{x}$ and support vectors $\mathbf{x}_s$ of the SVM as analog signals x(t) and respectively $x_s(t)$, which makes possible to describe the kernel evaluation in a simple analog signal processing fashion. The equivalence of the standard formulation and the proposed time-continuous formulation of the SVM classification phase is proven by building a Simulink model and by a set of experiments on the IRIS classification set. The graphical illustration of the result provided by the time-continuous representation of the signals is also useful to understand the SVM classification and to further reduce the computational time, providing an ordering of the features based on their significance can be found. In our future work we will implement the proposed solution with simple analog circuits and search interpolation methods to generate the continuous signals x(t) and $x_s(t)$.

THE AUTHORS

Gabriel Oltean and Mihaela Gordan are with the Basis of Electronics Department, Technical University of Cluj-Napoca,C. Daicoviciu 15, Cluj-Napoca, Romania.
E-mail: goltean@bel.utcluj.ro

REFERENCES

[1] I. Buciu, C. Kotropoulos, I. Pitas, "Combining support vector machines for accurate face detection", *IEEE Int. Conf. on Image Processing*, Thessaloniki, Greece, 2001, pp. 1054-1057

[2] V. P. Kumar, T. Poggio, "Learning-based approach to real time tracking and analysis of faces", *Proc. of AFGR*, 2000, France, 2000, pp. 96--101

[3] V. N. Vapnik, *Statistical Learning Theory,* J. Wiley, N.Y., 1998

[4] T. Joachims. *Making large-scale SVM learning practical. Advances in kernel methods - support vector learning*, B. Scoelkopf and C. Burges and A. Smola (ed.). MIT-Press, 1999.

[5] Ding Ai-ling, Liu Fang, Zhao Xiang-mo, "The massive data classifiers based on reduced set vectors method", *IEEE 2002 Int. Conf. on Comm., Circuits and Syst.,* vol. 2, pp. 1239 – 1242

[6] N. Ancona, G. Cicirelli, E. Stella and A. Distante, "Object detection in images: complexity reduction and parameter selection", *Proc. ICPR02*, vol. 2, pp. 426-429

[7] R. Genov, S. Chakrabartty, and G. Cauwenberghs "Silicon Support Vector Machine with On-Line Learning," *Int. J. of Pat. Recog. and Artificial Intelligence,* World Scientific, 2003, pp. 385-404

[8] R. A. Fisher, „The Use of Multiple Measurements in Axonomic Problems", *Annals of Eugenics 7*, 179-188, 1936

[9] S.R. Gunn, MATLAB Support Vector Machine Toolbox (Internet), March 1998