Published at: 16th INTERNATIONAL CONFERENCE ON
CONTROL SYSTEMS AND COMPUTER SCIENCE, Bucharest, 2007
http://cscs19.acs.pub.ro/files/Program_CSCS16_2007.pdf

A hardware oriented RBF classifier with a simple constructive training algorithm based on support vectors

Radu Dogaru, *Member, IEEE,*

***Abstract*—This paper introduces a compact and fast neural classifier, with a very simple hardware implementation yet having performances similar to those obtained using a support vector machine. The method of training introduced herein is simpler and more efficient than the typical SVM training methods. Our architecture is basically a LMS trained Adaline operating in a nonlinearly expanded feature space. The nonlinear expander is built upon a set of *m* hardware-oriented basis functions with centers selected as support vectors from the training set. The selection algorithm is a simple one, making the training complexity to be similar to that of training an Adaline i.e. O(N), where n is the number of samples.**

***Index Terms*— Neural networks, Signal classification, Support Vector Machines, VLSI.**

I. Introduction

A growing number of applications are developing in areas such as data mining, ubiquitous computing or ambient intelligence where certain pattern recognition tasks (e.g. large numbers of features characterizing a database, voice or biomedical signal or image recognition) are required to be performed at high speeds and with reasonable accuracy. Such systems are often regarded as VLSI-embedded neural processors where additional requirements of low power consumption and compactness are regarded.

While SVM networks, grounded by a very strong mathematical foundation, are regarded as being very accurate, it is known that they are also computationally intensive, especially during the training process. Moreover, their kernel functions need to satisfy a Mercer condition and therefore they are restricted to several widely known types only (e.g. the Gaussian kernel $K(\mathbf{x},\mathbf{c}) = \exp\left(-\gamma\|\mathbf{x}-\mathbf{c}\|\right)$. But hardware simplicity may demand simple kernels using no multipliers or other complicated mathematical operators. It is known that a SVM with the above kernel definition is equivalent with a radial basis network (RBF) with Gaussian radial basis functions. Although structurally the RBF and the SVM with Gaussian kernel are the same, there are certain differences. Usually, RBFs aim at training the centers and other parameters while in the SVM approach the centers are selected as a subset of the training set input samples, the so called “support vectors”. Training a RBF can be also an extensive and relatively complex task (e.g. training a RBF network with back-propagation). In [1] we proposed a different approach for training the RBF, introducing an architecture and a training algorithm called RBF-M (modified RBF) where new RBF neurons could be constructively created with centers selected from the training dataset. Then we were not aware on the SVM theory and in [1] the aim was to show that the newly proposed architecture can be used for certain regression problems (prediction of chaotic time-series). We were also able to show that the networks accuracy was not significantly affected by the choice of the RBF kernel and distance function. In particular, a simple to implement (hardware –oriented) RBF function:

$$\varphi^{tri}(d) = \begin{cases} 0 \text{ if } & d > k\cdot r \\ 1-\dfrac{|d|}{k\cdot r} & \text{else} \end{cases} \qquad (1)$$

was then defined as an approximation to the well known Gaussian:

$$\varphi^{gaus}(d) = \exp\left(-\frac{d^2}{2r^2}\right) \qquad (2)$$

and it was shown that the smallest error is obtained when approximating (2) with (1) if the coefficient k is chosen as $k = \sqrt{2\pi}$ (see Fig.1).

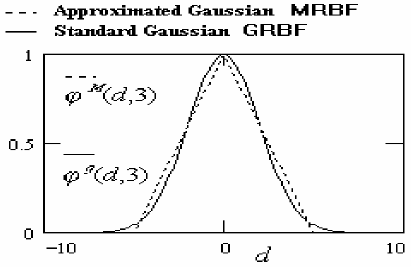

Fig.1. Gaussian and approximated (triangular) RBF

In the above, r represents the *radius* parameter (specific for any radial basis function), and $d = \|\mathbf{x}-\mathbf{c}_j\|$ is the *distance* between an input vector **x** and a center vector $\mathbf{c}_j$ ($j = 1,..,m$ is an index of the RBF neuron). Instead of the common Euclidian distance:

$$d^e = \|\mathbf{x}-\mathbf{c}_j\|_2 = \sum_{i=1}^{n}\left(x_i - c_{ji}\right)^2 \quad (3)$$

in [1] we demonstrated that a Manhattan distance

[footnote] Professor Radu Dogaru is with University “Politehnica” of Bucharest, Department of Applied Electronics (e-mail: radu_d@ieee.org, Tel/Fax: 0040-21-4024685). .

Published at: 16th INTERNATIONAL CONFERENCE ON
CONTROL SYSTEMS AND COMPUTER SCIENCE, Bucharest, 2007
http://cscs19.acs.pub.ro/files/Program_CSCS16_2007.pdf

(much easier to implement in hardware) can be used without changing in the overall performance:

$$d^m = \left\| \mathbf{x} - \mathbf{c}_j \right\|_1 = \sum_{i=1}^{n} \left| x_i - c_{ji} \right| \quad (3')$$

Within this paper we will reinvestigate the RBF-M introduced in [1] stressing on its equivalence with the SVM machine and showing that its simple constructive algorithm for finding the RBF units can be regarded as a straightforward method to identify *m* support vectors. This step of the training algorithm has a complexity of O(N) where N is the number of samples in the dataset. The reminder of the training algorithm is simply an Adaline trained in an *m* dimensional space formed by the outputs of the RBF neurons. Unlike SVMs, our method allows hardware oriented kernels of the type described by eqn. (1) and (3) cannot be used.

In Section 2, the general framework for training both the SVM and the RBF-M are presented with an emphasis on the similarities between the two architectures.

In Section 3 we have considered a wide set of benchmark classification problems for a comparison between the two architectures. It is shown that our method is as much as effective as the SVM with a much simpler training and with a better capability for a hardware implementation (e.g. FPGA or mixed-signal). Concluding remarks close our exposition.

II. SVM VERSUS RBF-M CLASSIFIERS.

For the sake of simplicity, in the following we describe the architectures and training algorithms for a simple (belonging or not belonging to class) classification problem. The extension to multi-class problems (with *M* categories) is done by employing *M* simple classifiers with their outputs connected to a winner-take-all subsystem ("one to rest classifiers" [2]). The classifier with the highest output value wins and thus indicates the recognized class.

We assume that a training set TR and a test set TS are available. The training set is used for updating the weights of the classifier while the test set is used to evaluate its generalization performance. Such sets can be constructed from a unique database by splitting it into sets with almost equal number of samples. A sample is defined as a pair $\left(\mathbf{x}^k, d^k\right)$, where $\mathbf{x}^k$ is an input vector with size *n* and d^k is a desired output $d^k \in \{-1,1\}$, $d^k = 1$ indicating that $\mathbf{x}^k$ belongs to the class. The training set contains *N* samples i.e. $TR = \{..,\left(\mathbf{x}^k, d^k\right),..\}$.

The SVM architecture and its training algorithm:

The output of a single SVM classifier[1] [3] is defined as:

$$y = b + \sum_{k=1}^{N} \alpha_k d^k K\left(\mathbf{x}, \mathbf{x}^k\right) \quad (4)$$

where *b,* and α_k are trainable parameters of the SVM algorithm. In the literature several training algorithm were proposed but all are known as being computationally complex. Specific for the SVM systems is the *data representation,* i.e. the training samples are explicitly included in the classifier equation. The *kernel* function $K\left(\mathbf{x}, \mathbf{x}^k\right)$ must satisfy the Mercer condition [3]. An usual choice for the kernel is the Gaussian or RBF kernel defined as:

$$K\left(\mathbf{x}, \mathbf{x}^k\right) = \exp\left(-\gamma \left\| \mathbf{x} - \mathbf{x}^k \right\|_2^{\,2}\right) \quad (5)$$

Such kernels can be also considered Radial Basis Functions using Euclidean distances (3), assuming that $r = 1 / \sqrt{2\gamma}$. However, note that RBF kernels such as the hardware oriented kernel proposed in [1] and described by (1) *cannot* be used within the SVM framework unless it is proved that they satisfy the Mercer's condition. The SVM training algorithm is a quadratic optimization problem with respect to the α_k trainable parameters. Different packages can be used, a detailed analysis being presented in [2] revealing a large computational complexity of $O(N^3)$. Recent decomposition methods [4] may reduce this complexity to $O\left(Q^3 N + nN^2 + \frac{nN^3}{Q}\right)$, but it still remains quite large. Some experimental results in [2] show that for a problem with N=2000 samples, a typical value of the number of operations in the training algorithm is of the order of at least 10^9. Assuming a covering value of *n*=100 inputs the above results (i.e. more than 5000 times *n*N*). This is equivalent with an Adaline trained for 5000 epochs !

The RBF-M classifier:

The RBF-M classifier [1] has an architecture identical with the SVM using RBF kernels. It is defined by the following equation:

$$y = w_0 + \sum_{k=1}^{m} w_k K\left(\mathbf{x}, \mathbf{x}^{j_k}\right) \quad (5)$$

where w_0 replaces b and w_k replaces $\alpha_k d^k$ are synapses of an Adaline [3] operating in an *m* dimensional output space of the *m* kernel functions $K\left(\mathbf{x}, \mathbf{x}^{j_k}\right)$, which can be defined as any combination of basis functions (1), (2) and distance $d = \left\| \mathbf{x} - \mathbf{x}^{j_k} \right\|$ defined by such formulae as (3) or (3')

[1] Here we will consider a continuously valued output. In a single class setup it should be interpreted according to its sign (a positive sign indicates membership to the class and a negative sign the opposite). In a multi-class system, a classifier is assigned to each category. During the retrieval phase the classifier with the largest value of its output wins and it determines the recognized class.

However, it has several advantages over the SVM:

i) The RBF kernel and the distance can be freely chosen as long as the general conditions for a radial basis function are satisfied [3]. Since we are interested in a hardware-efficient algorithm, in the next we will consider the kernel defined in (1), using the hardware oriented distance (3').

ii) The selection of *m* support vectors $\mathbf{x}^{j_k}$ is also done from the samples of the training set, but now this is done using a very simple algorithm called in the next a *selection algorithm* . The selection algorithm applies before training of w_k and requires only one epoch. The number *m* of support vectors results automatically from the selection algorithm.

iii) The computationally intensive training algorithm of the SVM is replaced by a simple LMS (or other similar methods, e.g. the Ho-Kashyap method in [5]) training. As shown in the last section, a number of 10 training epochs is usually sufficient, representing a reduction of more than 50 times in comparison to the SVM training!

The *selection algorithm* is as follows:

```
$m = 1, \quad k = 1, \quad j_k = 1$
// select the center of the first RBF unit as the first sample in the dataset.
$ov = 1;$ % an overlapping[2] coefficient is set to 1
FOR $j = 2,..N$ % for all training samples
// compute an activity level in the existent RBF units
    $act = \sum_{k=1}^{m} K\left(\mathbf{x}^{j}, \mathbf{x}^{j_k}\right)$
// create a new support vector (and RBF unit) only if the activity of the RBF is lower than the overlapping coefficient.
    IF $act < ov$
        $k = k + 1;$
        $j_k = j;$
        $m = m + 1;$
    END
END
```

The idea behind is simple: the RBF units "cover" a certain region from the input space. If an input vector appears which is not covered enough (i.e. the resultant activity of all RBF units is lower than *ov*) a new RBF unit shall be created and its associated center corresponds to the new input vector $\mathbf{x}^{j_k}$. As seen above, the algorithm provides a sequence of the integer indices j_k such that all RBF centers $\left\{\mathbf{x}^{j_k}\right\}_{k=1,..,m}$ are clearly specified among all input vectors from the training set. The number *m* of RBF units is determined automatically and it depends on the specific dataset TR, and on the *radius* parameter *r* of the RBF function defining the kernel used in (5). Although in [1] the algorithm allowed changing the overlapping parameter it was later found that a value of 1 is a good compromise allowing to reduce the tuning parameters to only one, i.e. the *radius* parameter.

The Adaline training

After one epoch applied to the above selection algorithm, the number *m* of the RBF units and their centers $\mathbf{x}^{j_k}$ are known so an Adaline with the inputs $z_k = K\left(\mathbf{x}\ , \mathbf{x}^{j_k}\right)$can be trained with the well known LMS algorithm [3]. Each synapse w_k it is adjusted with a quantity Δw_k after the presentation of a new input vector (on-line learning) according to the LMS learning law:

$$\Delta w_k = \eta z_k \left(d^k - \sum_{k=0}^{m} w_k z_k \right) \qquad (6)$$

In the above we assumed $z_0 = 1$ corresponding to the bias value w_0. Equation (6) is used for each sample in the training set until a training epoch is completed. After each training epoch, a test set is applied and the overall performance is calculated. Here we use PCIC defined as a *percentage of misclassified input vectors.* A fixed number of epochs *Ep* is considered, and the solution having the lowest PCIC during those epochs is kept.

There are two training modes:

a) **fast training:** This mode uses a training rate $\eta = 0.1$ and a single epoch *Ep=1.* This mode is used to allow a fast location of the best radius parameter r_best. Typical values for *r* range between 0.1 and 10. The following algorithm is used to detect the best radius:

```
Set r_min, r_max, delta_r
PCIC_best=100%
FOR r=r_min to r_max step delta_r
  * Apply the selection algorithm
  * Apply fast training of the Adaline
  IF PCIC<PCIC_best
        PCIC_best=PCIC
        r_best=r
  END
END
```

Typically it is enough to apply the above algorithm twice, first with a setup of the following kind: *r_min=0.1, delta_r=2, r_max=10.1 and then with a setup r_min=r_best-0.5, delta_r=0.2, r_max=r_best+0.5.* Therefore some 25x2=50 epochs (one for the selection algorithm and one for the fast

[2] It is a measure of the degree of overlap between two RBF units: a small degree (e.g. 0.1) implies that some input vector that is placed just in the middle of the distance between the two RBF centers will generate an overall output of the RBF output close to 0 making thus very difficult to discriminate such vectors. If the overlap becomes large (e.g. 10) the two RBF units are redundant.

training) suffice to locate the best value for the radius.

b) **Accurate training:** This mode is applied after determining the best radius r_best and it uses lower value of η and much more epochs (usually no more than 10 epochs are necessary) so that a much more accurate result is obtained.

From the above it follows that less than 100N iterations are required for the entire training algorithm to complete (including the finding of the optimal radius). Compared to the average 5000N reported for the SVM, there is a significantly lower computational effort in training the RBF-M while maintaining the support vectors idea. Note that in this comparison here we assumed that an optimal radius was found for the SVM. Actually this is a process with the same complexity as the one indicated above for the RBF-M, and thus training entirely (with tuning) the SVM leads to a computational effort even greater that the one considered above.

III. Experimental results and performance

In order to test and compare our architecture and evaluate its performances in comparison to the SVM an extensive set of benchmarks from [6] and [8] was used. A detailed description of each dataset can be found in [6][8]. Here we compare the performance of the RBF-M versus the classic SVM. For reference an SVM classifier called ‘OSUSVM’ (Matlab version 2.0) [7], which is freely available was used in our experiments.

The following table gives a comparison of the optimal performance for both classifiers (the gray shaded columns). Also the optimal value of the radius parameter and the resulting number of support vectors are also given.

Problem	*n*	*M*	**RBF-M**			**SVM**		
			Best PCIC (%)	**Best *r***	*m* RBF units	Best PCIC (%)	**Best *r***	*M* nr. of SVs
IRIS	**4**	**3**	0	1.4	5	4	1.29	32
WINE[3]	**13**	**3**	1.13	2.5	17	1.13	1.29	34
PHONEME[3]	**5**	**1**	21.5	1	11	22.5	0.35	236
OPT64	**64**	**10**	3.67	9	279	1.2	7.07	383
WAVE3	**21**	**3**	15.8	3.46	8	13.04	2.5	847
DIABETE	**8**	**1**	25.2	1.5	18	24.5	2.23	231
HEART	**35**	**1**	18.9	6.85	29	19.1	2.88	231
SATIMG[3]	**36**	**6**	11.87	2.75	117	8.95	0.31	1196

A differences in accuracy between the two methods of no more than ± 3% is observed. In some cases the RBF-M performs better, in other SVM performs better while in the remaining cases they obtained a similar accuracy. Therefore is hard to say that in terms of accuracy one method is better than the other. Note however that in all cases the SVM requires far more support vectors, even though in the table we count their average number (in fact there are different support vectors per each class, but in the table the average number per class is given). In the RBF-M, the support vectors *are the same* for all clases. This result shows that besides the much less complicated learning algorithm, the RBF-M architecture is more economical in terms of units. This is of course of great importance when dealing with hardware-oriented implementations but also useful in software implementations as well.

IV. Conclusions

A compact, VLSI tailored kernel-type neural architecture was proposed and its accuracy was proved to be comparable to the SVM. While the SVM is widely accepted as one of the most accurate trainable classifier it is shown here that by using a different, simpler approach to generate support vectors, a novel architecture can be defined, with a much more economical implementation and less computation during the training process while maintaining the accuracy (measured here as percentage of misclassification).

The architecture allows hardware-oriented kernels requiring no more than summation and absolute value operators but other classic kernels can be used as well while there is no more a need to test for the Mercer condition.

Further research will focus on the extensive use of the RBF-M architecture in various applications, in the form of FPGA or micro-controller based modules.

References

[1] Dogaru, A.T. Murgan, S. Ortmann, M. Glesner, “A modified RBF neural network for efficient current-mode VLSI implementation”, in *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems (Micro-Neuro’96)*, IEEE Computer-Press, Laussane 12-14 Febr. 1996, pp. 265-270, 1996.

[2] K. K. Chin, **“**Support Vector Machines applied to Speech Pattern Classification”, MPhil. Dissertation, Darwin College, 1998. *Available from: http://svr-www.eng.cam.ac.uk/~kkc21/thesis_main/thesis_main.html*

[3] Haykin, S., *Neural networks: A Comprehensive Foundation –second edition*, 1999, Prentice Hall.

[4] Osuna, E., Girosi, F., “An improved training algorithm for support vector machines*”, Proceedings of the 1997 NNSP IEEE Workshop (Neural Networks for Signal Processing),* pp. 276-285.

[5] Hassoun, M.H., *Fundamentals of Artificial Neural Networks*, MIT Press, 1995.

[6] Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases *http://www.ics.uci.edu/~mlearn/MLRepository.html Irvine,*

[3] From “ELENA” database [8].

CA: University of California, Department of Information and Computer Science.

[7] Junshui Ma , “OSU Support Vector Machines (SVMs) Toolbox, version 2.00, Oct. 2000”, available from: available from
http://www.ece.osu.edu/~maj/osu_svm/

[8] ELENA database: http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/databases/