ADAPTIVE NONLINEAR FILTERING WITH THE SUPPORT VECTOR METHOD

*Davide Mattera** *Francesco Palmieri** *Simon Haykin***

* Università degli Studi di Napoli Federico II, Dipartimento di Ingegneria Elettronica e delle Telecomunicazioni, Via Claudio, 21, I-80125 Napoli, ITALY
e-mail: mattera@diesun.die.unina.it
** McMaster University, Communications Research Laboratory, 1280 Main St. W., Hamilton, Ontario, CANADA L8S 4K1

ABSTRACT

The recently introduced Support Vector Method (SVM) is one of the most powerful methods for training a Radial Basis Function (RBF) filter in a batch mode. This paper proposes a modification of this method for on-line adaptation of the filter parameters on a block-by-block basis. The proposed method requires a limited number of computations and compares well with other adaptive RBF filters.

1 INTRODUCTION

RBF filters constitute an important class of nonlinear filters [1] that can be used with success for describing the relationships between input and output measurements of a large variety of practical systems encountered in diverse fields of science and engineering such as communications, structural engineering, fluid mechanics and so on.

Adaptive filtering can be viewed as the problem of adaptively modeling an unknown time-varying system. Let us suppose that the value at time step n of an output time series $x(\cdot)$ depends, by means of an unknown (and, possibly, time-varying) deterministic relation F, called the target function, on the pseudo-state vector $\boldsymbol{r}(n) \triangleq [x(n-1), \ldots, x(n-p), u(n), u(n-1), \ldots, u(n-q)]^T$.

The filter aims at approximating the unknown dependence F on the base of the input and output time-series measurements. The input-output mapping performed by the RBF filter is accomplished in two stages: 1) a nonlinear transformation which maps the input space onto an intermediate space; 2) a linear transformation, which maps the intermediate space onto the output. Therefore, the filter output $\hat{x}(n)$ at time n is equal to:

$$\hat{x}(n) = \sum_{k=1}^{K} w_k \exp\left[-(\boldsymbol{r}(n) - \boldsymbol{t}_k)^T \boldsymbol{C}_k (\boldsymbol{r}(n) - \boldsymbol{t}_k)\right] \quad , \tag{1}$$

where $\{K, w_k, \boldsymbol{C}_k, \boldsymbol{t}_k\}$ is the set of filter parameters. The input-output measurements $\{u(n), x(n)\}_{n=1,\ldots,L}$ can be organized into a set of pairs, called examples, $\{(\boldsymbol{r}(n), x(n))\}_{n=1,\ldots,l}$. and the training process aims at determining a value for the filter parameters such that expression (1) closely approximates the unknown target function F.

Many methods have been proposed in the literature [1] to determine the parameter values; many of them are composed of two stages: in the former one, the value[1] of $\boldsymbol{C}_k$ and $\boldsymbol{t}_k$ are determined by some unsupervised clustering algorithm; in the latter one, the coefficients w_k are determined in accordance with a mean-square error criterion. The main drawback of these methods is constituted by the fact that the former stage is usually not performed in accordance to some optimality criterion but it is mostly based on heuristics. Moreover, the number K of centers that assures a good approximation is, essentially, considered known *a priori* and it is, in practice, determined by means of some cross-validation criterion.

Very recently, a new method, called Support Vector Method, deeply rooted in Statistical Learning Theory [5], has been developed for obtaining an approximation to an unknown function from a set of given examples. The SVM constitutes an approximate implementation of the structural risk minimization [5], an inductive criterion which aims at minimizing an upper bound on the generalization error of the filter, rather than minimizing only the mean square error over the given examples. The first experimental results of a comparison study with the RBF method have shown [3] that SVM achieves better performance with respect to the standard RBF method.

All these methods, however, in their typical formulation, use a batch mode of computation as they calculate an approximation of the unknown target function using the examples derived from a predetermined block of observations. For example, should one decide to use a larger data sample or to dynamically add new samples as they become available on-line, one should discard the previous estimate and calculate a new one from the beginning. Since the filter output $\hat{x}(n)$ in (1) depends linearly on the weights w_k, one could apply the clas-

[1] It is assumed often that $\boldsymbol{C}_k \triangleq \frac{1}{\sigma_k^2}\boldsymbol{I}$ or even that $\sigma_k = \sigma$ for all k.

sic methods of linear filtering [4, 1] to update the second stage. However, the other filter parameters , in the first RBF stage, cannot be, so easily, adapted with these methods. Gradient descent techniques can be applied, but the rigorous formulation of SVM may be lost since convergence may remain unpredictable and the number of centers has to remain fixed. An interesting class of RBF methods (see references in [6]) has been proposed for determining all the filter parameters with an extended Kalman type algorithm. An interesting version of these methods, which does not need *a priori* knowledge about the number of centers K, has been proposed in [6].

In this paper, a modification of the basic SVM applicable in a time-varying scenario is proposed. The SVM approximation is adaptively modified in a block-by-block basis. The number of steps between successive re-trainings is not fixed; in particular, the filter is re-trained only if it does not satisfy the requirement that the approximation error is smaller than a given parameter ε. In each re-training the previous structure is not completely discarded, but its support vectors are kept and added to the new set of examples from which the new filter structure is obtained. This allows to keep limited the necessary computational effort. A performance analysis of this simple method is carried out by means of computer simulations on the well-known Lorenz data [2]; a comparison with another adaptive RBF method [6] shows superior performance.

2 THE SUPPORT VECTOR METHOD

The method aims at achieving an ε-approximation to the unknown target function, i.e. the approximation function is required to be within an ε-mask around the unknown target function. Given the set of l examples $\{\boldsymbol{r}_i, y_i\}_{i=1,\ldots,l}$, the approximation has the following form:

$$\hat{x}(n) = \sum_{i=1}^{l} c_i \mathcal{K}(\boldsymbol{r}(n), \boldsymbol{r}_i) + b \ , \tag{2}$$

where $\mathcal{K}$ is a fixed function that has to satisfy Mercer's condition:

$$\int\int \mathcal{K}(\boldsymbol{u}, \boldsymbol{v}) g(\boldsymbol{u}) g(\boldsymbol{v}) d\boldsymbol{u} d\boldsymbol{v} > 0 \ , \tag{3}$$

for all $g \neq 0$ for which $\int_{\mathcal{R}} g^2(\boldsymbol{u}) d\boldsymbol{u} < \infty$. A possible choice for $\mathcal{K}$ that satisfies condition (3) is the Gaussian kernel with fixed variance σ^2

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{1}{\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2) \quad . \tag{4}$$

The linear coefficients c_i in (2) are set equal to $\alpha_i^* - \alpha_i$ and obtained as solution of the following optimization problem [5]:

$$\max_{\alpha_i^*, \alpha_i} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)\ y_i - \sum_{i=1}^{l}\varepsilon(\alpha_i^* + \alpha_i) \ , \tag{5}$$

subject to the condition:

$$\sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0 \ , \tag{6}$$

and, such that, for each $i \in \{1, \ldots, l\}$,

$$0 \leq \alpha_i^* \leq C, \qquad 0 \leq \alpha_i^* \leq C \ , \tag{7}$$

where C is a fixed parameter and ε defines the required accuracy. At the solution only some values of α_i^* and α_i are different from zero; the training examples $\boldsymbol{r}_i$, such that $\alpha_i^* - \alpha_i$ is different from zero, are included in the representation (2) and are named *support vectors* [5]. The structural complexity of the obtained filter and, also, the computational requirement of the training process depend, mainly, on the number S of support vectors.

The constant b in (2) is determined by the following relation:

$$b = y_i - \left[\sum_{j=1}^{l} c_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] - \varepsilon \ sign(c_i) \ , \tag{8}$$

valid for any i such that $0 < c_i < C$.

3 THE ADAPTIVE SVM (ASVM)

The tracking of the output of an unknown filter can be solved by the basic SVM and by periodic re-trainings of the filter structure. If the batch SVM has to be utilized, at each re-training one needs to repeat all the computations on an increasing example set. Clearly the computational load could become quickly unaffordable. A strict batch application of the basic algorithm in a periodic re-training mode may also be inappropriate to track suddenly changing systems because the past memory of the method has to be appropriately selected. Moreover, one needs to specify when perform re-training and account for the computational complexity.

The development of an adaptive version of the SVM is based on the following property: the optimum solution S_l provided by SVM on a set E_l of l examples is also the optimum S_{l+1} on a new set of $l+1$ examples, obtained as union of E_l and of a new example, if the absolute value of the error of the solution S_l on the new example is smaller than ε. Therefore, re-training of the filter is done only if the filter output does not satisfy the required ε-approximation, i.e. if, for, say, N consecutive time steps, the error, averaged over the last, say, L examples,

$$\mathcal{E}(n) = \frac{1}{L}\sum_{i=1}^{L}(\hat{x}(n-i) - x(n-i))^2 \tag{9}$$

exceeds the required value ε^2.

The simplification of the training algorithm is obtained by utilizing as re-training set only the union of the Support Vectors of the old filter and the new N examples on which the old filter exhibits error larger than ε^2. The optimality of the algorithm compensates the reduction in the size of the training set necessary to maintain low the computational requirements.

The rationale behind this method is clearly the fact that the support vectors are really what carries the most relevant information about the past data to solve the specific mapping problem.

We summarize here the different steps of the method:
1) The algorithm is initialized from the first s examples, which are obtained from the first $s+max(p,q)$ values of $(u(n),x(n))$. Suppose now that the computation time required to get the optimal parameters on these s examples is T_0.
2) The corresponding filter F_0 with Support Vectors S_0 is therefore available at time step $s+max(p,q)+T_0+1 \triangleq l_0+1$. Let us call $l_{0,1}, l_{0,2}, \ldots, l_{0,N}$ the consecutive time steps, successive to l_0, on which the average approximation error $\mathcal{E}(n)$ is larger than ε^2.
3) At time step $l_{0,N}$ re-training starts on the training set TR_1 constituted by the union of the set S_0 and the N examples at time instants $l_{0,j} \quad j=1,\ldots,N$. Let us name T_1 the training time, S_1 the correspondent set of Support Vectors and F_1 the solution. The filter F_0 is therefore utilized from the instant l_0+1 to the instant $l_{0,N}+T_1 \triangleq l_1$, the filter F_1 from the instant l_1+1, and so on.

For a fixed training set size, the computational requirements of the training algorithm is mainly dependent on the number of support vectors at the solution. This number depends on the required accuracy which is controlled by the approximation parameter ε. Therefore, the value of ε allows to trade off the accuracy of the obtained filter with the required computational complexity.

The proposed method is naturally suited to a scenario in which a highly-nonlinear target function varies with a time-horizon of quasi-stationarity sufficiently long. However, a correct choice of ε allows to obtain good performance in relation to the existing methods also on scenarios with shorter horizon of quasi-stationarity. An interesting property of the proposed method is the minimization of the number of Gaussian kernels in the approximation filter needed to obtain the required accuracy; its absolute value depends on the complexity of the unknown filter structure.

4 SIMULATION RESULTS

In this section the results of a performance analysis of the proposed method, carried out by computer simulations, are presented and compared with that of the adaptive RBF method proposed in [6]. The time-series utilized in our experiments is the output of the well known Lorenz system [2]:

$$\begin{aligned} \dot{x} &= \rho(y-x) \\ \dot{y} &= -x\ z + rx - y \\ \dot{z} &= x\ y - b\ z \end{aligned} \qquad , \tag{10}$$

with the following values of the parameters: $\rho = 16$, $b = 4$ and $r = 45.92$. The set of equation (10) are integrated with a fourth-order Runge-Kutta method and generated 10000 samples at 16-bit resolution with a sampling frequency of 40 Hz are generated. The time-series is then normalized so that its mean-square value is equal to one. Suppose that only the first component x is observed and the pseudo-state vector is $\boldsymbol{r}(n) \triangleq (x(n-1),\ldots,x(n-p))^T$ with $p=10$ and $q=0$. The free parameters in the SVM are three: C, σ^2 and ε; C and σ^2 are chosen equal to 3 and 7.5 respectively while for ε, that controls the required accuracy, the two values 0.1 and 0.01 have been considered; moreover, $N=10$ and $L=30$. Performance analysis is carried out by simulating $J=20$ different realizations of the whole process by integrating (10) with 20 different starting points. The squared error at time n in the jth realization is computed as

$$\Gamma_j(n) \triangleq \frac{1}{M}\sum_{m=1}^{M}(\hat{x}_j(n+m)-x_j(n+m))^2 \tag{11}$$

with $M=300$, where $x_j(\cdot)$ is the true time-series for the jth trial, $\{x_j(n+m), m=1,\ldots,M\}$ are the outputs of the filter with parameters frozen at the value they had at time n and with the true pseudo-state vector $\boldsymbol{r}_j(n)$ at the input; the performance is, then, evaluated through the mean-square error (MSE) $\Gamma(n) \triangleq \frac{1}{J}\sum_{j=1}^{J}\Gamma_j(n)$.

Fig. 1 shows both the MSE in dB of the classic batch SVM and of the ASVM for Lorenz data for two different values of required accuracy $\varepsilon=-20dB$ ($\varepsilon=0.1$) and $\varepsilon=-40dB$ ($\varepsilon=0.01$). One can notice that the convergence of the adaptive filter is comparable with that of the optimal batch filtering algorithm which performs total re-training at each new step.

The required number of support vectors, at time step $n=2000$, is quite similar in the two cases. In fact, for ASVM it is equal to 22 and to 104 for $\varepsilon=-20dB$ and $\varepsilon=-40dB$ respectively, while in the batch SVM it is equal to 34 and 148 respectively. The ASVM, therefore, can also be seen as a method for reducing the complexity of the SVM, without affecting too much the approximation accuracy.

A comparison of the convergence performance of the ASVM with one of the best adaptive RBF method is also presented. The considered algorithm, Minimal Resource Allocation Network (MRAN), introduced in [6], is based on extended Kalman filtering and has been shown in [7] to obtain good accuracy with a simpler structure when compared to the classical multi-layer perceptron.

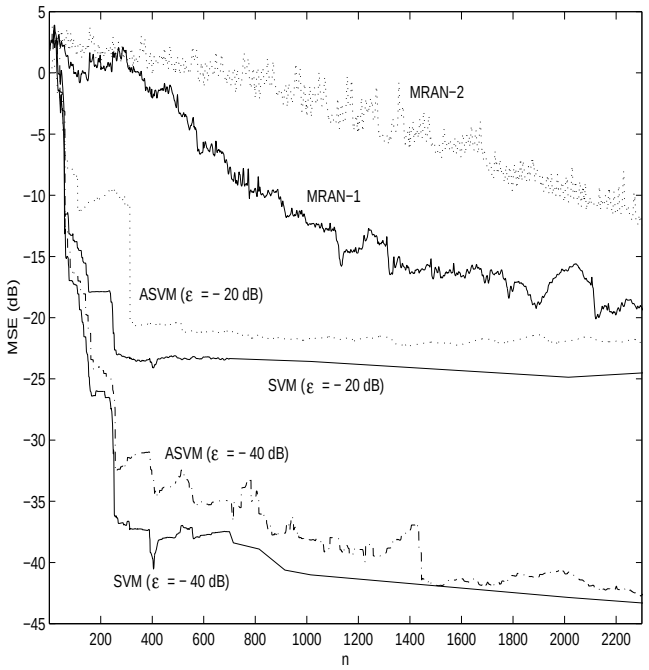

Figure 1: The MSE in dB for the two best choices (named MRAN-1 and MRAN-2) of MRAN parameters and for the SVM and ASVM for $\varepsilon = -20$ dB ($\varepsilon = 0.1$) and $\varepsilon = -40$ dB.

Fig. 1 shows the MSE for MRAN with reference to the same Lorenz time-series. The ASVM method exhibits faster convergence and better accuracy with respect to MRAN.

Another advantage of the ASVM consists in the fact that the number of free parameters[2] is limited to two (C and σ). Conversely, the MRAN method requires the setting of nine free parameters and it is not immediately clear how their choice influences performances. More specifically, it is not known in which way one can trade off the accuracy of the approximation with the complexity of the obtained RBF. This trade off is controlled in the SVM by means of the parameter ε. Fig. 1 shows the results corresponding to our best two choices for the MRAN parameters.

The computational complexity of the training process for the ASVM is weakly dependent on the dimension p of the pseudo-state vector $\boldsymbol{r}(n)$ while it depends strongly on the number of Support Vectors that are needed to represent the target function within the required ε-approximation. The complexity of the training process for the MRAN is instead proportional to $(K \cdot p)^2$ where p is the dimension of the pseudo-state vector $\boldsymbol{r}(n)$ and K is the number of centers. This number is automatically selected by the algorithm and can vary at each step since the algorithm can add new centers or prune them out as their linear weight remains very small for a given number of steps.

The simulation experiments have also shown that, sometimes, for certain choices of the free parameters, the number of centers, required by the MRAN before convergence, become so large to render the training time[3] much larger than the one required by the ASVM. However, the simulations also show that the average number of centers required at convergence by MRAN is generally smaller than that required by the ASVM. This is the main disadvantage of the ASVM and it is due to its inability to adapt the size of each center.

5 CONCLUSIONS

In this paper a modified version of the SVM has been introduced for adaptive filtering. In the simulation experiments performed on a well-known chaotic time series, the proposed method has shown superior performance with respect to other techniques based on extended Kalman filtering, both in terms of accuracy and speed of convergence. The preliminary results presented here render this method quite promising for nonlinear adaptive filtering applications.

REFERENCES

[1] S. Haykin, *Adaptive filter theory*, III ed., Mc-Graw Hill, 1996.

[2] E.N. Lorenz, "Deterministic nonperiodic flow", J. Atmos. Sci., vol. 20, pp. 130-141, 1963.

[3] B. Scholkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, "Comparing support vector machines with Gaussian kernels with radial basis function classifiers", IEEE Trans. on Signal Processing, vol. 45, no. 11, pp. 2758-2765, 1997.

[4] J. Stark, "Recursive prediction of chaotic time series", Journal of Nonlinear Science, vol. 3, pp. 197-223, 1993.

[5] V.N. Vapnik, *The nature of the statistical learning theory,* Springer-Verlag, 1995.

[6] L. Yingwey, N. Sundararajan, P. Saratchandran, "Identification of time-varying nonlinear systems using minimal radial basis function neural networks", IEE-Proc. Control Theory Appl., vol. 144, no. 2, March 1997.

[7] L. Yingwey, N. Sundararajan, P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm", IEEE Trans. on Neural Networks, vol. 9. no. 2, pp. 308-318, March 1998.

[2] Strictly speaking, also N, L and ε can be considered free parameters in the ASVM. However, their role is directly related to the desired performance and not to the unknown target function.

[3] The simulation experiments were performed in the assumption that the time required to execute the calculations needed at each step were sufficient small to render possible the on-line implementation of both methods.