Available online at www.sciencedirect.com

Future Generation Computer Systems 21 (2005) 1056–1067

FGCS Future Generation Computer Systems

www.elsevier.com/locate/fgcs

Vector quantization: a weighted version for time-series forecasting

A. Lendasse[a, *], D. Francois[b], V. Wertz[b], M. Verleysen[c]

[a] *Computer Science Division, University of Memphis, CND, 335A FedEx Institute of Technology, Memphis, TN, USA*
[b] *CESAME, Université catholique de Louvain, av. G. Lemaitre, 4, B-1348 Louvain-la-Neuve, Belgium*
[c] *DICE, Université catholique de Louvain, pl. du Levant 3, B-1348 Louvain-la-Neuve, Belgium*

Available online 9 April 2004

Abstract

Nonlinear time-series prediction offers potential performance increases compared to linear models. Nevertheless, the enhanced complexity and computation time often prohibits an efficient use of nonlinear tools. In this paper, we present a simple nonlinear procedure for time-series forecasting, based on the use of vector quantization techniques; the values to predict are considered as missing data, and the vector quantization methods are shown to be compatible with such missing data. This method offers an alternative to more complex prediction tools, while maintaining reasonable complexity and computation time.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Time-series prediction; Vector quantization; Missing data; Radial-Basis Function Networks

1. Introduction

Time-series prediction encompasses a wide variety of application domains, in engineering (electrical load estimation), environmental (river flood forecasting), finance (risk prediction, and other domains. The problem consists in estimating an unknown time-dependent variable given its past history, and possibly exogenous context variables.

To predict time series, a large variety of approaches are available. The most widely used ones are certainly the linear models, i.e. those estimating the unknown value as a linear combination of other variables. The latter are not necessarily restricted to past values of the series and exogenous variables: they may also include estimations of the past predictions, or errors on them for example. This class of models, including AR, ARMA, ARX, ARMAX, Box-Jenkins, etc. has the advantage of being quite simple to use, well recognized, and does not suffer too much from the choice of structural parameters. Unfortunately, there are a number of real-world situations where a simple linear model on the data does not apply. The prediction results obtained by a linear method may thus reveal poor or simply useless results.

Nonlinear prediction models may overcome this limitation, by dropping the linear model hypothesis on the data. Nonlinear prediction tools include artificial neural networks (Multi-Layer Perceptrons,

* Corresponding author.
E-mail addresses: lendasse@cnd.memphis.edu, lendasse@auto.ucl.ac.be (A. Lendasse), francois@auto.ucl.ac.be (D. Francois), wertz@auto.ucl.ac.be (V. Wertz), verleysen@dice.ucl.ac.be (M. Verleysen).

Radial-Basis Function Networks, Self-Organizing Maps, etc.), fuzzy logic models, and many others. Despite their more general character, and therefore their extended potential, nonlinear tools suffer from other drawbacks: they usually rely on complex optimization procedures, they request the difficult setting of many user-controlled structural parameters (usually controlling complexity), etc. In most situations, the level of performances reached with nonlinear models depends on the experience of the user. Moreover, as the extensive use of cross-validation or other resampling techniques for model selection may strongly influence the performances, this level also depends on computational resources and afforded computation time!

In this paper, a forecasting method based on nonlinear tools with limited complexity is presented, as a compromise between the two approaches. The limited complexity allows adjusting the structural parameters when applicable, by sweeping the range of possible values with extended cross-validation. The method is based on vector quantization (VQ) techniques, for which efficient learning procedures are known. It is shown that VQ algorithms can easily accommodate missing data; this property is used by considering the values to predict as missing data in a generalization step of the method, while learning is performed on past values of the series, i.e. nonmissing data. Smoothing of the results is optionally performed by using Radial-Basis Function Networks (RBFNs).

In the following, it is first described how nonlinear tools in general, and RBFN in particular, may be used in the context of time-series prediction. Vector quantization is then introduced, together with its use on problems with missing data. The prediction method is then detailed, with optional improvements implementing different compromises between possible performances and complexity. These compromises are illustrated on two time-series prediction examples. Finally, the computational load of the methods is studied in terms of number of operations.

2. Linear and nonlinear models

The simplest models that may be used for time-series prediction are the Autoregressive (AR) ones. In these models, the prediction y_t is computed as a linear combination of some previous values of the time series:

$$\hat{y}_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n}. \tag{1}$$

The regressor is the vector containing the past values $(y_{t-1}, \ldots, y_{t-n})$ of the series. The size n of the regressor must be fixed a priori.

Other linear models also use exogenous variables: the Autoregressive models with eXogenous variables (ARX). In these models, some external data are added to the regressor:

$$\begin{aligned}\hat{y}_t = {} & a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} \\ & + b_1 u_{t-1} + b_2 u_{t-2} + \cdots b_{nb} u_{t-nb}\end{aligned} \tag{2}$$

with u_t a time series of exogenous data. Of course, more than one exogenous time series could be added to Eq. (2).

Other linear models that can be used are the linear Autoregressive models with Moving Average and eXogenous variables (ARMAX models). In these models, the prediction error generated during the previous prediction is added to the regressor:

$$\begin{aligned}\hat{y}_t = {} & a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} \\ & + b_1 u_{t-1} + b_2 u_{t-2} + \cdots + b_{nb} u_{t-nb} \\ & + c_1 r_{t-1} + c_2 r_{t-2} + \cdots + c_{nc} r_{t-nc}\end{aligned} \tag{3}$$

with

$$r_t = y_t - \hat{y}_t. \tag{4}$$

If the prediction obtained by linear models is not accurate enough, nonlinear models can be used. The linear models presented above can be extended to nonlinear ones. For example, the Nonlinear Autoregressive models (NAR models) are defined as

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-n}, \theta), \tag{5}$$

where f is a nonlinear function and θ the parameters of this function. An example of nonlinear model is the Radial-Basis Function Networks, schematically represented in Fig. 1.

The output of the RBFN is given by

$$\hat{y} = \sum_{i=1}^{M} \lambda_i \Phi_i(x, C_i, \sigma_i), \tag{6}$$

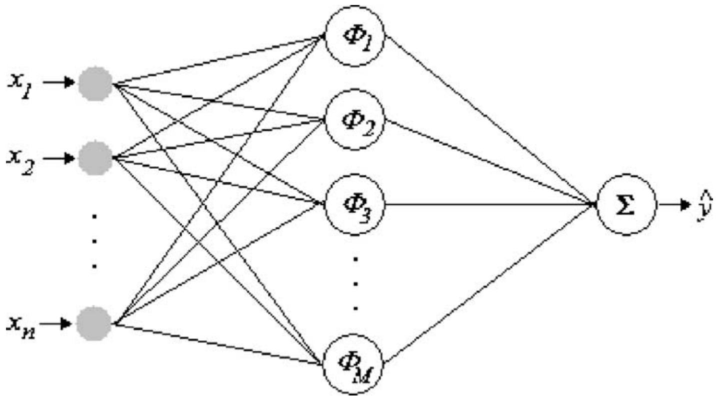

Fig. 1. RBFN with n inputs and one output.

where

$$\Phi_i(x, C_i, \sigma_i) = \mathrm{e}^{-(\|x-C_i\|/\sqrt{2}\sigma_i)^2}. \tag{7}$$

In Eqs. (6) and (7), x represents the vector of inputs $x_1, \ldots, x_n$, and $\hat{y}$ the output of the model. In our NAR model (Eq. (5)), the inputs x_i are the previous value of the series (the regressor $[y_{t-1}, \ldots, y_{t-n}]$). The parameters of the model are the centers C_i (usually resulting from vector quantization of the input data), the standard deviations σ_i, and the multiplying coefficients λ_i. Learning of RBFN parameters may be performed through various learning algorithms (see for example [1,9–12]); what is important to mention here is that an RBFN usually requires the adjustment of at least two structural parameters: the number of Gaussian kernels and at least one parameter—overlapping factor—needed for the adjustment of the standard deviations σ_i (see for example [1]). In our time-series prediction context, the vector inputs x are replaced by the regressors. An example of RBFN result on a one-dimensional problem is shown in Fig. 2.

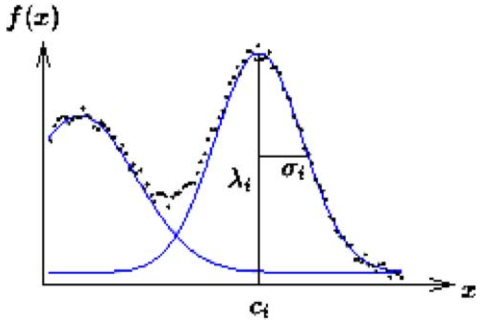

Fig. 2. Example of RBFN. Dots: learning data; solid lines: the two Gaussian functions in the model used to fit the data.

Model selection is an important part in the process of learning with a nonlinear model. In the case of RBFN for example, there is no rule to select a priori an optimal number M of Gaussian functions; the number of Gaussian functions is a structural parameter as discussed above. In order to choose optimal values for structural parameters, validation procedures are used: the learning set is used to determine the parameters of the model, such as the a_i coefficients for the linear models or C_i, σ_i and λ_i for an RBFN, while the validation set is used to determine the structural parameters such as the size of the regressor or the number of Gaussian functions in an RBFN.

In the learning phase the parameters are determined as the result of the minimization of the prediction error on the learning set LS. The Normalized Mean Square Error on the learning set can be defined as [14]

$$\mathrm{NMSE_L} = \frac{\sum_{t \in \mathrm{LS}}(y_t - \hat{y}_t)^2}{\sum_{t \in \mathrm{LS}}(y_t - \overline{y})^2}, \tag{8}$$

where $\overline{y}$ is the average value of y_t. During learning, the structural parameters are fixed.

There are two important differences between linear and nonlinear models with respect to learning:

1. The parameters of AR and ARX models are determined directly and exactly (for example with a pseudo-inverse). On the contrary parameters of nonlinear models are determined iteratively (for

example with a gradient descent method). The consequence is that the parameters obtained for nonlinear models are often not optimal because the iterative methods usually lead to local minima.
2. The computational load for determining the parameters of nonlinear models is huge compared to the computational load in the linear case.

In the validation phase, the structural parameters are determined as the result of the minimization of the prediction error on a validation set (usually, the minimization is simply performed by sweeping the structural parameters over the discrete range of possible values). The Normalized Mean Square Error on the validation set VS can be defined as

$$\mathrm{NMSE_V} = \frac{\sum_{t \in \mathrm{VS}} (y_t - \hat{y}_t)^2}{\sum_{t \in \mathrm{VS}} (y_t - \bar{y})^2}, \tag{9}$$

where $\bar{y}$ is the average value of y_t. Using a validation set independent from the learning set reduces the risk of choosing structural parameters leading to overfitting.

3. Vector quantization

3.1. *Definition*

Vector quantization [3] is a way to summarize the information contained in a large database. Vector quantization is the basic tool used in the prediction method described in the next section; it is also used as part of RBFN learning [1].

Let us define a database through the example of Table 1.

Elements (observations, or data, or vectors) of this database are thus lines $V_i (1 \leq i \leq N)$, with elements $V_{ij} (1 \leq j \leq D)$. The principle of vector quantization is to replace this database by another one, containing fewer vectors C_k of the same dimension $D (1 \leq k \leq M, M < N)$. The number M of vectors C_k is a structural parameter of the method. The new vectors C_k, called centroids, are deemed to have a distribution similar to the distribution of the initial V_i. Once the centroids are fixed through some algorithm, each vector V_i is then quantized by its “nearest-neighbor” centroid C^*:

$$C^*(V_i) = \arg\min_{C_k} (||V_i - C_k||^2). \tag{10}$$

The Voronoi region associated to centroid C_k is the set of vectors associated to C_k by Eq. (10). In other words, it is the region of the space nearest to C_k than to any other centroid.

Determining the “best” set of centroids, i.e. the set that minimizes on average (on all data contained in the original database), the quantization error defined by the distance in Eq. (10), is a nonlinear optimization problem. Many algorithms have been proposed in the literature to determine the centroids: batch algorithms, as Lloyd’s one [8], or on-line adaptive ones, as the competitive learning method [6].

As an example, competitive learning works as follows. At each iteration, a data V_i is drawn randomly from the initial database. The nearest centroid C^* is determined (according to Eq. (10)) and moved (adapted) in the direction of data V_i:

$$C^{*\,\mathrm{new}}(V_i) = C^{*\,\mathrm{old}}(V_i) + \alpha(V_i - C^{*\,\mathrm{old}}(V_i)), \tag{11}$$

where α is a time-decreasing adaptation factor. The same operation is repeated several times (epochs) over each vector of the whole database.

Of course, all VQ algorithms find a set of centroids that correspond to a local minimum of the quantization

Table 1
Database definition

	Variable 1	Variable 2	...	Variable j	...	Variable D
First data	11	17	...	V_{1j}	...	87
Second data	12	34	...	V_{2j}	...	7
Third data	82	−32	...	V_{3j}	...	92
Fourth data	34	65	...	V_{4j}	...	42
...	...	...	...	...	...	...
ith data	V_{i1}	V_i2	...	V_{ij}	...	V_{iD}
...			...	...	...	...
Nth data	−2	34	...	V_{Nj}	...	V_{ND}

Table 2
Database with missing data

	Variable 1	Variable 2	...	Variable j	...	Variable D
First data	11	17	...	V_{1j}	...	87
Second data	?	34	...	?	...	7
Third data	82	?	...	V_{3j}	...	92
Fourth data	34	65	...	V_{4j}	...	?
...	...	...	...	...	...	...
ith data	V_{i1}	V_{i2}	...	V_{ij}	...	V_{iD}
...			...	...	...	...
Nth data	−2	?	...	V_{Nj}	...	V_{ND}

error criterion, and not a global one. Many initialization methods and many adaptation rules have been proposed in the literature (see for example [3,6]), aiming at finding a good compromise between the “quality” of the local minimum and the computational complexity (number of operations).

3.2. Missing data

An easy-to-understand but not so well known and used property of VQ is that databases with missing data can be quantized easily. Let us imagine that the initial database is now as represented in Table 2.

In Table 2, each ‘?’ corresponds to a value that was not measured, or not stored, for any reason; the ‘?’s are the missing values. The VQ process defined by Eqs. (10) and (11) remains valid, if the computation of Euclidean distance in Eq. (10) is replaced by a distance computed only on the known variables and not on the missing ones (of course, it is assumed that the centroids have no missing values). Similarly, the adaptation of centroids by Eq. (11) is performed only using the corresponding existing coordinates of data V_i, and not on the missing ones.

Using VQ with missing data has proved to be efficient to find estimations of missing values in databases [4]. A missing value is replaced by the corresponding variable of the centroid nearest to the observation. This technique may be viewed as estimating a missing value with a function of the available variables in the same observation:

$$\hat{V}_{ij} = (C^*(V_i))_j, \tag{12}$$

where C^* is the nearest centroid from V_i according to Eq. (10) where the distance computation is restricted to the known coordinates of V_i. Using the known information from V_i is certainly more appropriate than replacing a missing value by the average of the corresponding variable from all other observations (average on a column of the database), as it is often the case when dealing with incomplete data. VQ may be viewed as a simple but efficient way to use the information contained in the known coordinates of vector V_i.

4. Prediction with VQ and missing data

4.1. Quantization without weighting

Time-series prediction may be viewed as a regression problem when expressed by Eq. (5), repeated here for convenience:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-n}, \theta). \tag{13}$$

In this equation, $\hat{y}_t$ is the estimation of the unknown value of the time series at time t, while $y_{t-1}, \ldots, y_{t-n}$ are the known n past values of the series which form the regressor. θ is the set of parameters of the estimator f; the estimator can be linear or nonlinear in the general case. The problem is now expressed as a regression problem with n inputs and one output instead of a time-series problem.

The past knowledge on the series is concatenated with the output to form a $(n+1)$-dimensional vector of the following form:

$$Y_t = \begin{bmatrix} y_t & y_{t-1} & y_{t-2} & \cdots & y_{t-n} \end{bmatrix}. \tag{14}$$

The vectors Y_t may be thought of as the vectors V_i from the database presented in the previous section. VQ can thus be applied to those vectors resulting from the concatenation. Next, the VQ model may be used on incomplete vectors, where the missing value is precisely the

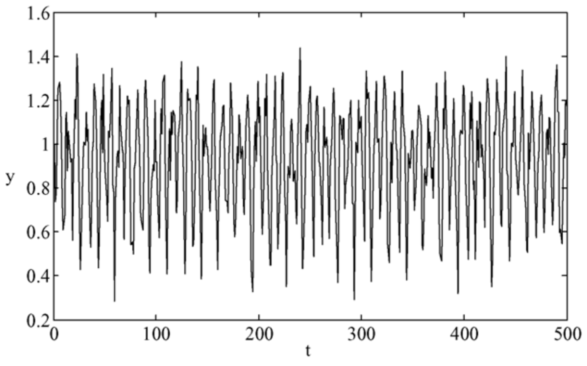

Fig. 3. Mackey–Glass series, with additive noise (see text for parameters).

y_t one. Therefore, model *f* in Eq. (13) consists in the approximation of the missing value by the corresponding coordinate of the nearest centroid, as given by Eq. (10).

This method is illustrated in this paper on a well-known time-series benchmark, the Mackey–Glass series [5]. The series has been generated according to

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \beta y(t) + \frac{\alpha y(t-\delta)}{1+y(t-\delta)^{10}} \tag{15}$$

with $\beta = -1$, $\alpha = 0.2$ and $\delta = 17$. The series is sampled with a sampling period of 6. A part of the series is illustrated in Fig. 3. Ten thousand points have been generated; the first 5000 ones are used for learning (VQ), the 5000 last ones for validation. Each regressor contains the four last values of the series, as recommended in the literature [13]. Gaussian noise with 0.25 variance has been added. The results are given in Fig. 4. The optimal number of centroids is 20, and the optimal NMSE is found to be 0.30. For the sake of comparison, a linear model using the same regressor gives NMSE = 0.36 (using the same learning and validation sets).

If we now use the previous time series but without any added noise (Fig. 5), we observe the surprising result that the optimal number of centroids is equal to the number of data in the learning set (Fig. 6).

This result is not so surprising when considered the fact that data generated by Eq. (15) are noiseless. With so many centroids, the VQ method becomes equivalent to lazy learning [2], a method which consists in looking in the learning data the one closest from the regressor. Lazy learning has proved to be efficient in noiseless problems.

4.2. Quantization with weighting of the output

The method illustrated in the previous section has an obvious limitation: if the size of the regressor is large, the relative weight given to the y_t predictions in the quantization of vectors Y_t (Eq. (14)) decreases (it counts for one coordinate, with regards to *n* coordinates for the regressor).

A straightforward extension of the method is then to give a different weight to the quantization of the y_t predictions by building the Y_t vectors as follows:

$$Y_t = [\, k y_t \quad y_{t-1} \quad y_{t-2} \quad \cdots \quad y_{t-n}\,]. \tag{16}$$

Increasing *k* means that the VQ is biased to give more importance to the prediction. The structural parameter *k* may be determined in the same way as the number *M* of centroids in the VQ: by optimization on

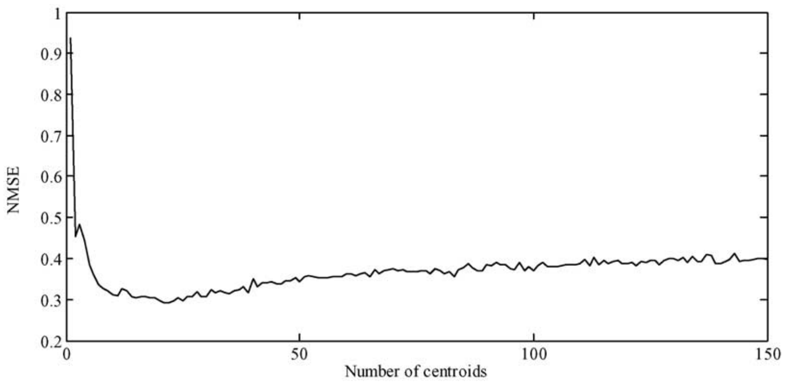

Fig. 4. NMSE on the validation set for the Mackey–Glass with additive noise series prediction with VQ.

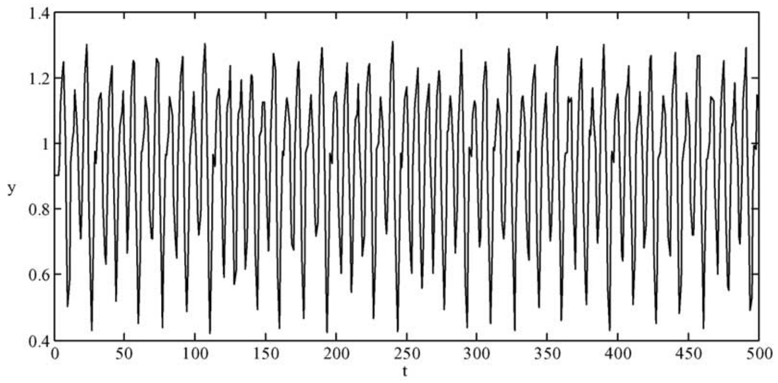

Fig. 5. Mackey–Glass series, without noise (see text for parameters).

a validation set. Nevertheless, as the optimization is realized in practice by scanning the range of parameters, a double optimization can be time-consuming. It is therefore suggested to optimize the two parameters consecutively, possibly in an iterative way. This way of optimizing the parameters does not guarantee to find a global minimum for both of them, but is a reasonable compromise between the quality of the optimization and the computational load.

On the noisy Mackey–Glass series, starting with $M = 20$ (the result found in the previous section for $k = 1$), an optimization on k gives $k = 0.5$. This value is then used for another optimization on M, which results in $M = 60$ (see Fig. 7), leading to a decreased 0.27 NMSE. Further optimizations do not improve the result anymore.

4.3. *Quantization with weighting of the inputs by a linear model*

The VQ process (12) implements a particular nonlinear model as defined by Eq. (13). Nevertheless, compared to other nonlinear models, such as Multi-Layer Perceptrons (MLPs), Eq. (12) has the drawback that all inputs (all coordinates of Y_t vectors (14) but y_t) are weighted equally. This is of course not optimal. A further step is thus to weight also the inputs, building the Y_t vectors as follows:

$$Y_t = [\,k y_t \quad k_1 y_{t-1} \quad k_2 y_{t-2} \quad \cdots \quad k_{n-1} y_{t-n}\,]. \qquad (17)$$

However, the optimization of all k_i parameters is heavy. A full search is often not practicable, while a gradient descent (computed by finite differences on the

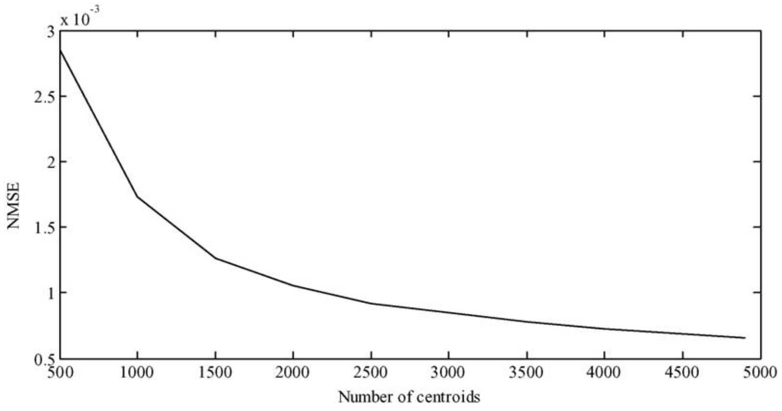

Fig. 6. NMSE on the validation set for the Mackey–Glass series prediction with VQ.

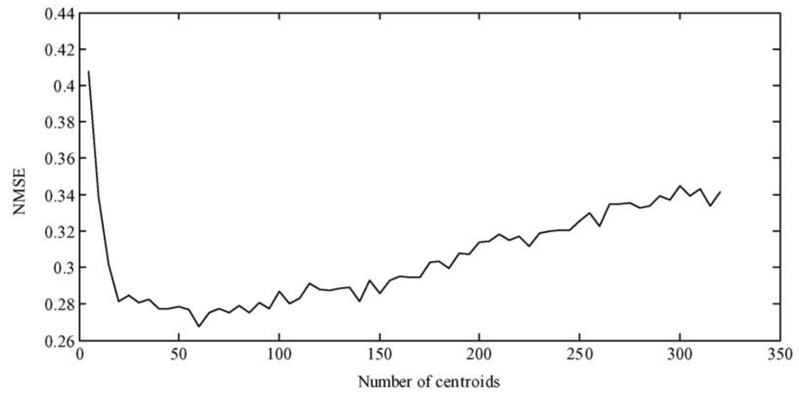

Fig. 7. NMSE on the validation set for the Mackey–Glass with additive noise series prediction with weighted VQ, $k = 0.5$.

validation set results) may be very slow. For this reason, we suggest the use of the weights of a linear model, as explained below.

When building a linear model:

$$\hat{y}_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_{n-1} y_{t-n}, \qquad (18)$$

the a_i coefficients may be considered as the importance (weighting) that variable y_{t-i} has on the output. In other words, it is the first partial derivative of the output with respect to a specific input. The a_i coefficients can therefore be considered as a first-order approximation of the k_i coefficients needed in (17). Of course, the a_i resulting from a linear hypothesis will not be optimum in the sense defined in the previous section. Nevertheless, again, we are looking for a good compromise between an impracticable full search and an inefficient a priori choice such as $k_i = 1$. The coefficients given by linear model (18) appear to be efficient regarding this compromise.

A linear model with a four-dimensional regressor on our previous example, the noisy Mackey–Glass series, gives roughly identical a_i coefficients. As a consequence, the use of these coefficients does not improve the previously presented results. To illustrate the weighting of the VQ by the coefficients of a linear model, we thus use another well-known time-series prediction benchmark, the SantaFe A series. Noise with variance = 80 has been added to the original series. A part of this series is illustrated in Fig. 8. As for the previous example, 5000 points have been used for learning and 5000 for validation. The regressor size is 6. Fig. 9 shows the NMSE resulting from the

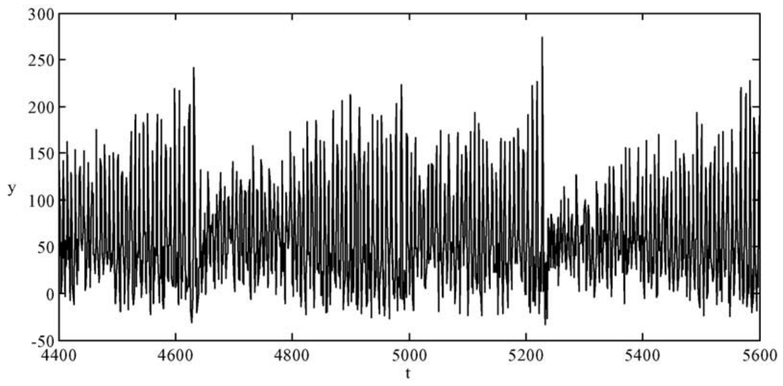

Fig. 8. SantaFe A series.

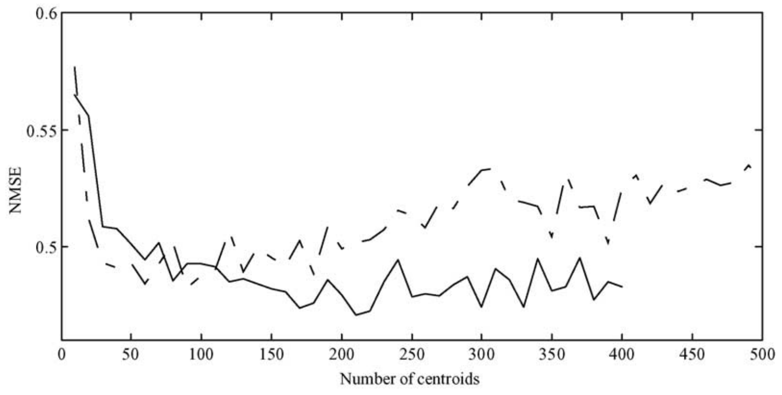

Fig. 9. NMSE on the validation set obtained on the prediction of the noisy SantaFe A series with a VQ model; dashed: without weighting of the inputs; solid: with weighting of the inputs by the coefficient of a linear model.

weighting with the coefficients of a linear model; the improvement is slight (NMSE equal to 0.46 instead of 0.48), but does not demand large computation. Furthermore, the NMSE obtained by weighting the inputs leads to a flatter curve, and so finding an optimal number of centroids is not so crucial. The *k* weighting of the output y_t has been obtained with the method described in Section 4.2.

4.4. Radial-Basis Function Networks with weighted inputs

With VQ, the possible values for the prediction form a finite set (the set of outputs in the centroids); the output may thus be viewed as quantized. We might want to smooth the output for example by placing Gaussian kernels on each centroid. In that way, we merely construct an RBF Network.

The principle of weighting the inputs, presented in the context of VQ, may be applied to RBFN [7] too. Weighting the output in an RBFN model has no sense, as appropriate output weights are computed through the optimization of λ_i, which is a standard procedure in RBFN learning. However, RBFN suffer from the same drawback as VQ what concerns the a priori identical weighting given to all inputs. In conventional RBFN context, this is illustrated by the use of a scalar σ_i standard deviation, instead of a variance/covariance matrix.

Fig. 10 shows the results of an RBFN model trained on the noisy SantaFe A series; inputs have

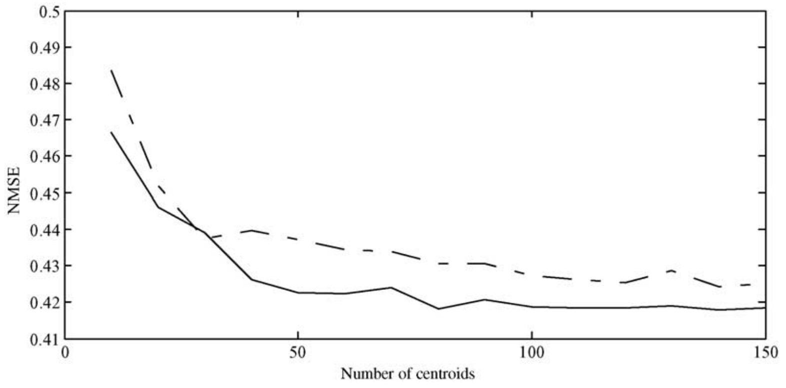

Fig. 10. NMSE on the validation set obtained for the prediction of the noisy SantaFe A series with an RBFN model; dashed: without weighting of the inputs; solid: with weighting of the inputs by the coefficients of a linear model.

Table 3
Complexity and number of operations of each method

Method	NMSE	Complexity			Number of operations proportional to
		Structural parameters	Iterative fitting		
			Number of parameters	Objective function	
Linear	0.7	0	–	–	$N \times n$
VQ	0.48	1	$M \times n$	Simple	$N \times M \times \#domM \times n \times E$
VQ + weight out	0.46	2	$M \times n$	Simple	$N \times M \times \#domM \times n \times E \times K$
VQ + weight in	0.45	2	$M \times n$	Simple	$N \times M \times \#domM \times n \times E \times K$
RBFN	0.43	2	$M \times n$	Simple	$N \times M \times \#domM \times n \times E \times S$
RBFN + weight in	0.42	2	$M \times n$	Simple	$N \times M \times \#domM \times n \times E \times S$

been weighted according to the coefficients of a linear model. The training of the RBFN model has been realized according to [1]. The NMSE is 0.42. The same remarks about weighting for the VQ technique apply here.

5. Discussion

In this section, we summarize the results obtained with the methods presented above, and compare their inherent complexities.

We will first recall that in our problem, only learning is time-consuming. Once the chosen model is built, using it with a fresh piece of data is very fast and does not demand heavy computations. We assume that complexity of an optimization algorithm only depends on the inherent complexity of the objective function and on the number of parameters to estimate. Noniterative tasks are negligible compared to iterative tasks; in addition, optimizing a structural parameter demands doing the entire job several times to find its optimum. In our problems, the optimal regressor size is supposed to be known and fixed.

Table 3 shows the complexity and number of operations for all methods presented in this paper applied on the SantaFe data. In this table:

- *domM* is the domain of tested values for the number of centroids (VQ and RBFN).
- M is the average value in *domM*.
- *#domM* is the number of elements in *domM*.
- n is the size of the regressors.
- N is the number of data in the learning set.
- E is the number of epochs in the VQ learning process.
- K is the size of the set of tested values for k.
- S is the size of the set of tested values for the standard deviation in the Gaussian kernels.

The structural parameters are the number of centroids or Gaussian kernels (all methods), the parameter weighting the output in VQ-based methods, and the width factor in RBFN-based ones; their impact on the number of operations is respectively *#domM*, K and S.

In comparison, other nonlinear models such as RBFN with other learning methods and Multi-Layer Perceptrons, have at least two structural parameters, leading to a number of operations at least equivalent to the largest one in Table 3. In addition however, their objective function is much more complex (for example back-propagation for MLP); the number of operations should be multiplied by the relative complexity of the objective function.

We can see in Table 3 that VQ techniques give much better performances than linear models, at a cost which is not much higher, and in any case much lower than if using a fully nonlinear model.

6. Conclusion

The paper introduces a new time-series prediction method based on vector quantization. The idea is to gather past values into regressors and concatenate them with the desired output that is the value of the series one step ahead. Vector quantization is then applied on those vectors, resulting in the determination of a set of centroids which can be seen as patterns to identify in the time series. When prediction of a new value has to be made, the missing coordinate of the appropriate centroid is used. Adequately weighting the elements of the vectors and using Gaussian kernels as improvements of the method have shown to give improved results. The method is illustrated here on time-series prediction problems, but can be extended to any regression task.

The method presented in this paper implements a compromise between the complexity of learning and the performances of the method. Other nonlinear tools with complex, time-consuming learning algorithms might still result in better performances. However, complex learning combined with an appropriate choice of structural parameters through extensive cross-validation may become too costly in a wide variety of applications; at the other extreme, linear models often lead to poor performances. The method presented in this paper may therefore help practitioners to benefit from the performances of nonlinear models while maintaining reasonable complexity and computation times.

Acknowledgments

The work of A. Lendasse is supported by NASA Research Grant No. NCC-2-1244. Michel Verleysen is Senior research associate at the Belgian National Fund for Scientific Research (FNRS). The work of D. François and V. Wertz is supported by the Interuniversity Attraction Poles (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

References

[1] N. Benoudjit, C. Archambeau, A. Lendasse, L. Lee, M. Verleysen, Width optimization of the Gaussian kernels in radial basis function networks, in: Proceedings of the European Symposium on Artificial Neural Networks, ESANN 2002, Bruges, Belgium, 2002, pp. 425–432.
[2] G.L. Bontempi, M. Birattari, H. Bersini, Lazy learning for local modeling and control design, Int. J. Contr. 72 (7/8) (1998) 643–658.
[3] R. Gray, Vector quantization, IEEE Mag. 1 (1984) 4–29.
[4] S. Ibbou, Classification, analyse des correspondances et methodes neuronales, Ph.D. Thesis, Université de Paris 1 Sorbonne, Paris, 1998.
[5] M.C. Mackey, L. Glass, Oscillations and chaos in physiological control systems, Science 197 (1977) 287–289.
[6] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, vol. 30, 3rd ed., Springer, Berlin, 2001.
[7] A. Lendasse, J. Lee, E. de Bodt, V. Wertz, M. Verleysen, Approximation by radial-basis function networks—application to option pricing, in: Proceedings of the ACSEG 2002 on Connectionist Approaches in Economics and Management Sciences, Boulogne-sur-Mer, France, 2002, pp. 201–212.
[8] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (1982) 127–135.
[9] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 281–294.
[10] T. Poggio, F. Girosi, Networks for approximation and learning, Proc. IEEE 78 (1987) 1481–1497.
[11] M. Powell, Radial basis functions for multivariable interpolation: a review, in: J.C. Mason, M.G. Cox (Eds.), Algorithms for Approximation, Clarendon Press, New York, NY, USA, 1987, pp. 143–167.
[12] M. Verleysen, K. Hlavackova, Learning in RBF networks, in: Proceedings of the International Conference on Neural Networks (ICNN'96), Special Sessions Volume, Washington, DC, USA, 1996, pp. 199–204.
[13] J. Vesanto, Using the SOM and local models in time series prediction, in: Proceedings of the WSOM'97: Workshop on Self-Organizing Maps, 1997.
[14] A.S. Weigend, N.A. Gershenfeld, Times Series Prediction: Forecasting the Future and Understanding the Past, Addison-Wesley, Reading, Massachusetts, USA, 1994.

A. Lendasse was born in 1972 in Tournai, Belgium. He received the MS degree in mechanical engineering from the Universite catholique de Louvain (Belgium) in 1996 and the MS in control from the same University in 1997. He received his PhD degree in 2003, in the Centre for Systems Engineering and Applied Mechanics (CESAME) in the Universite catholique de Louvain and he was teaching assistant in the Applied Mathematics Department of this university from September 1998 to October 2003. He is now a postdoctoral researcher in the Computational Neurodynamics Laboratory, Computer Science Division at the University of Memphis. This position is founded by the NASA Grant NCC-2-1244. He is author or co-author of about 40 scientific papers in international journals, books or communications to conferences with reviewing committee. His research concerns time series prediction, Kohonen maps, nonlinear projections, nonlinear approximators, KIII models and EEG Classification. His homepage is http://cnd.memphis.edu/∼lendasse/.

D. Francois was born in Namur, Belgium, in 1979. He received the MS degree in computer science from the Université catholique de Louvain (Belgium). He is now a research assistant and PhD student at the Center for System Engineering and Applied Mechanics (CESAME) of the same university. His work is supported by the Belgian FRIA and concerns high-dimensional data analysis, nonlinear regression and classification, variable selection.

V. Wertz was born in Liège, Belgium in 1955. He got his engineering degree in applied mathematics in 1978 and a PhD in control engineering in 1982, both from the catholic University of Louvain. He is now professor at the catholic University of Louvain, in Louvain-la-Neuve, Belgium, after having held a permanent position with the NFSR. His main research interests are in the fields of identification, predictive control, fuzzy control and industrial applications. Lately, he has also been involved in a major pedagogical reform of the first and second year teaching program of the school of engineering.

M. Verleysen was born in 1965 in Belgium. He received the MS and PhD degrees in electrical engineering from the Université catholique de Louvain (Belgium) in 1987 and 1992, respectively. He was an invited professor at the Swiss EPFL (Ecole Polytechnique Fédérale de Lausanne, Switzerland) in 1992, at the Université d'Evry Val d'Essonne (France) in 2001, and at the Université Paris I-Panthéon-Sorbonne in 2002 and 2003. He is now a senior research associate of the Belgian FNRS (Fonds National de la Recherche Scientique) and Lecturer at the Université catholique de Louvain. He is editor-in-chief of the Neural Processing Letters journal and chairman of the annual ESANN conference (European Symposium on Artificial Neural Networks). He is author or co-author of about 130 scientific papers in international journals and books or communications to conferences with reviewing committee. He is the co-author of the scientific popularization book on artificial neural networks in the series “Que Sais-Je?”, in French. His research interests include artificial neural networks, self-organization, time-series forecasting, nonlinear statistics, and electronic implementations of neural and biomedical systems.