Vector Quantization Algorithms for One-Dimensional and Two-Dimensional Time Series

Svetla Radeva, Dimitar Radev and Vladimir Yakov

Abstract: *The purpose of this investigation is determining the parameters of non-stationary models for different kind of non-stationary processes. With the help of neural network and vector quantization is suggested an approach for modelling one-dimensional and two-dimensional time series. The parameters of vector quantization are determined for describing probability density function. Programs on MATLAB 6.5 and Visual C++ were created for modelling of time series and receiving their probability density function. These models can be used for describing and analyses the seismic waves behaviour, for queuing models in communication systems and for different kind of models of time series with long-range dependences.*

Key words: *Non-stationary Stochastic Processes, Artificial Intelligence, Vector Quantization, Neural Networks, Communication Systems*

INTRODUCTION

Modelling of non-stationary stochastic processes is an approach for stochastic analysis of a dataset, presented as time-series. The stochastic analysis gives possibility to receive different characteristics about time-series behaviour, which can be used for evaluation of the processes, presented with time-series [1, 2].

The classical approach for modeling stochastic processes with their probabilistic characteristics can be combined with artificial intelligence modelling. Implementation of neural networks for modelling of time series can be combined with stochastic and numerical methods. In this way can be obtained any desired accuracy for estimation of non-stationary processes parameters. The accuracy of received results is increasing because of learning capability of neural networks.

For the purposes of neural modeling the basic parameters of stochastic model are given as input parameters for structured according certain rules neural network, which should be in advance learned to generate input values according to given target classes. This learning is realizing with Learning Vector Quantization (LVQ), which can be implemented for:

- prognoses of stochastic processes;
- modeling of long-range dependant stochastic processes;
- evaluation of probability density distribution of random time-series;
- modeling of discrete and continuous processes, described with time-series etc.

The basic problem at vector quantization is how to learn the neural network in such a manner, that the received results of analysis and received evaluated parameters are describing efficiently the process under investigation. For realizing the process of vector quantization the classes according to which have to be realizes learning, calling target classes should be given in advance. When there is big variety of data, determining of target classes is very important. This determining is time consuming and it is necessary to develop algorithms for automation of their adjusting.

VECTOR QUANTIZATION PARAMETERS

With vector quantization can be presented the probability distribution function of time series, which can be used for determining of density distribution of time series. Time series are one-dimensional, when they are considering in time and two-dimensional, when there are two different input time series. For determining the entrance distribution it is necessary to adjust the probability density function, which can be made with vector quantization. The input information for neural network is treating as fluid flow of one or two time series. For these purposes is using a two-layered neural network, where the first layer is competitive and the second – linear. In competitive layer the input flow is transferring into module of vector quantization, where for each target class the neural network determine a winner.

As criteria is using some kind of distance (Euclidean, Manhattan or other) between target classes d_i, where $i = 1,\ldots,M$ is the index of target class. The cell with smallest value is "winner" and her output value is equal to 1, while values of the rest cells are equal to 0. For each value of input time series the weight matrix generates weight coefficients $w_{i,k}$ with the help of which is calculating distance d_i.

In linear layer the boundaries between target classes are adjusting and as a result is receiving the probability density function. After learning to recognize patterns, the neural network is training certain numbers of epochs for receiving more precise values for probability density function.

The Kohonen learning rule [1] is used when the winning node represents the same class as a new training pattern. The difference in class between the winning node and a training pattern causes the node to move away from training pattern by the same distance. In training, the winning node of the network, which is nearest node in the input space to a given training pattern, moves towards that training pattern, while dragging with its neighbouring nodes in the network topology. This leads to a smooth distribution of the network topology in a non-linear subspace of the training data. The goal of a learning neural model for vector quantification is to determine the probability density function for entrance distribution and each steady state of one- or two-dimensional time series.

ONE-DIMENSIONAL VECTOR QUANTIZATION

At one-dimensional vector quantization on the input of neural model is given one-dimensional input vector with n values. The input samples should be classified into different target classes.

The proper defining of target classes plays a basic role at founding a precise solution about density distribution of entrance values of time series. The task is to determine M target classes, which boundaries are unknown in advance. The vector quantization gives possibility for searching solution of this task. Here can be considered two cases:

- target classes have equally width and different number of target values in each class;
- target classes have equally number of target values in each class and different width of each class.

Both cases are applying in one-dimensional vector quantization, but second one gives more precise results. Difficulties in second case are connected with adjusting of boundaries between classes in such a manner that in each class is equal number of target values.

TWO-DIMENSIONAL VECTOR QUANTIZATION

At two-dimensional vector quantization on the input of neural model is given two-dimensional input vector. The entrance distribution of the input attribute space is determined with two neural layers – first, competitive, based on a set of input/target pairs, presented on (1) for N-dimensional input vectors,

$$\{\mathbf{x}_1,\mathbf{C}_1\},\{\mathbf{x}_2,\mathbf{C}_2\},\ldots,\{\mathbf{x}_j,\mathbf{C}_j\},\ldots,\{\mathbf{x}_N,\mathbf{C}_N\} \qquad (1)$$

with the help of which is trained the neural network at second, linear layer. Here $\mathbf{x}_j$ are two N-dimensional input vectors, and the M- dimensional vector $\mathbf{C}_j$ describes the condition of target classes, presented at (2).

$$\begin{aligned} \mathbf{x}_j &= \{X_j^{(1)}, X_j^{(2)}\}, & j &= 1,\ldots,N \\ \mathbf{C}_j &= \{S_1, S_2, \ldots S_k, \ldots S_M\}, & k &= 1,\ldots M \end{aligned} \qquad (2)$$

The hidden neurons from first layer compete via initialising of the weight matrix $\mathbf{W}_{kj}$ and are determining the winner. This is the neuron, which has minimal Euclid distance d_k to the input vectors $\mathbf{x}_j$.

Then the corresponding target class receives value 1 and the rest target classes receive 0, as is presented by (3).

$$\begin{aligned} &S_k = 1, \quad for \quad d_k^{\min}, \qquad d_k = \sum_{j=1}^{N}\left(X_j - W_{kj}\right)^2 \\ &S_k = 0, \quad otherwise \end{aligned} \qquad `(3)$$

The neuron-winner has feedback negative links to the rest of neurons and strong positive link to himself, which is used for learning in linear layer. During the training in the next epoch *q* are changing the coefficients of all neurons according to Kohonen learning rule [1], which is summarized at (4).

$$W_{kj}(q) = W_{kj}(q-1) \pm \xi\left(X_j(q) - W_{kj}(q-1)\right), \quad 0 < \xi \le 1 \qquad (4)$$

The coefficient ξ depends on the number of training epochs *q* and can be adjusted in advance in interval [0,1], where standard is determined equal to 0,1. The sign before the training coefficient ξ is positive for the neuron-winner, and negative for the neighbour neurons. As a result during the process of the training is changed the area of neighbour neurons for the neuron-winner, e.g. decreases the Euclid distances.

In two-dimensional output space is expected a map, corresponding to the k - dimensional array of output neurons C_j, which can be one or two-dimensional. The connection between *n*-dimensional input vector and *k*-dimensional output neural vector is realized with the weight matrix **W**. At competitive learning for winner is selected the output neuron j^*, which weight vector is closer to the current input according to (5).

$$\left|W_{jm}^* - X_m\right| \le \left|W_{jm} - X_m\right| \qquad \forall j \in [1,\ldots,k], \forall m \in [1,\ldots,n] \qquad (5)$$

The Kohonen learning rule is determined by (6).

$$\Delta W_{jm} = \xi \wedge \left(j, j^*\right)\left(X_m - W_{jm}\right) \qquad (6)$$

The neighbourhood function $\wedge(j, j^*)$ is equal to 1 if $j=j^*$, and decreases with increasing of distance between neurons j and j^* in input space. The neurons closer to the winner j^*, changes their weights more quick than remote neurons, for which the neighbourhood function is very small. The learning rule (6) attracts the winner's weight vector to the point X_m.

ALGORITHMS FOR VECTOR QUANTIZATION

There exists two approaches for both, one and two-dimensional vector quantization. They are connected with number of target values in each class and with equally or not equally classes width. That's why here we define four algorithms for these four cases.

Algorithm 1: one - dimensional vector quantization for equally width of the target classes.

1. Define the length of the interval (divide bigger and smaller value) and after it divide the received result into number of classes.
2. With Vector Quantization is determining class's structure: for each class is determining total number of target values, which is different.
3. Calculating of entrance distribution for each class.
4. Determining of weight vectors for all classes.
5. Learning and training the neural network for determining the target values via minimum-squared-error.
6. Recalculating of the target values in each class and density distribution.

Algorithm 2: one - dimensional vector quantization for equally numbers of target values in each class.

1. Determining the minimal and maximal value of time series, which will be left boundary of the first class and right boundary of the last class, respectively.
2. Divide number of values into number of target classes for receiving equal number of target values in each class.
3. Determining of class's structure: sort all values ascending and determine belonging of each value to certain class.
4. Determining of class's boundaries.
5. Determining of weight vectors for all classes via learning and training.
6. Calculating of entrance distribution for each class.
7. Adjusting boundaries of the classes for better performance of entrance distribution if necessary.
8. Recalculating of the target values in each class and if necessary go to step 6.

Algorithm 3: two - dimensional vector quantization for equally width of classes.

1. Define the width of each class first on axis X and than on axis Y as in Algorithm 1.
2. For each class described as rectangular zone are determining total number of target values, which is different.
3. Determining of weight vectors for all classes via learning and training.
4. Calculating of entrance distribution for each class.

Algorithm 4: two - dimensional vector quantization for equally numbers of target values in each class.

1. The user determine number of target classes on axis X as n_X and than on Y as n_Y.
2. Determine the class's structure: horizontal boundaries on axis X as in Algorithm 2 and than on axis Y in such a manner to receive in each class $n/(n_X \times n_Y)$ values, where n is total number of values. For each element of certain class on axis X, determine vertical boundaries via sorting ascending on Y the values from corresponding class.
3. Calculating of weigh centres for each class on the base of target values into it.
4. Determining of weight vectors for all classes via learning and training.
5. Calculating of entrance distribution for each class and adjusting boundaries of the classes for better performance of entrance distribution if necessary.

MODELLING OF ONE-DIMENSIONAL TIME-SERIES

Consider one-dimensional time series, where, for example are taken values from accelerogram, registrated at Alfero, Italy on October 25, 1985. The accelerogram is a typical one without first phase and it duration of is about 25 seconds, where the values of one-dimensional time series are recorded on every 0,002 seconds.

Under consideration are first 6 second, where are registrated about 3000 values of time series on axis North-South. On Figure 1 are presented the original time series (upper – left) and afterward are given the received values, when the input space are divided into different number of classes.

First on Figure 1 are given time series values, when the input space is divided into 50 classes and there are 2% smoothing (upper – right). Afterward are given cases when the input space of one-dimensional time series is divided into 40 classes and there are 2,5% smoothing (down – left) and when the input space is divided into 20 classes and there are 5% smoothing (down – right). The results are obtained with Algorithm 2 where one - dimensional vector quantization has equally numbers of target values in each class.

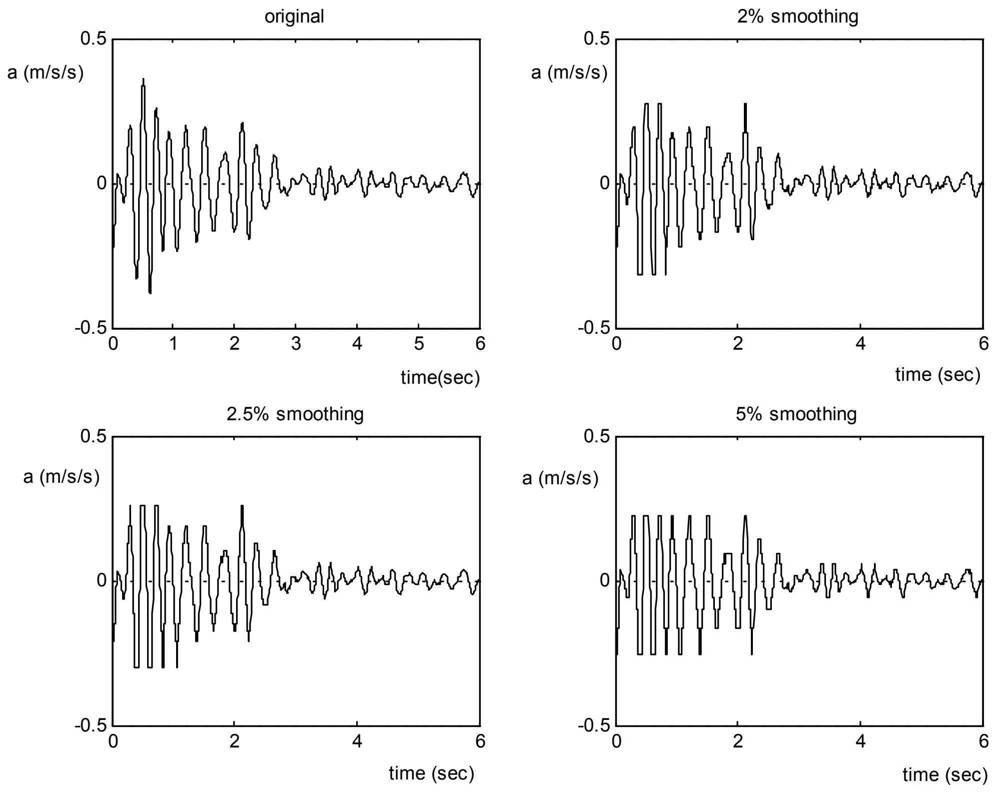

Figure 1. One-dimensional time series with different number of classes

On Figure 2 is given the probability density function (pdf), when one-dimensional time series is divided into 40 classes and there are 2,5% smoothing (for Figure 1 - down – left).

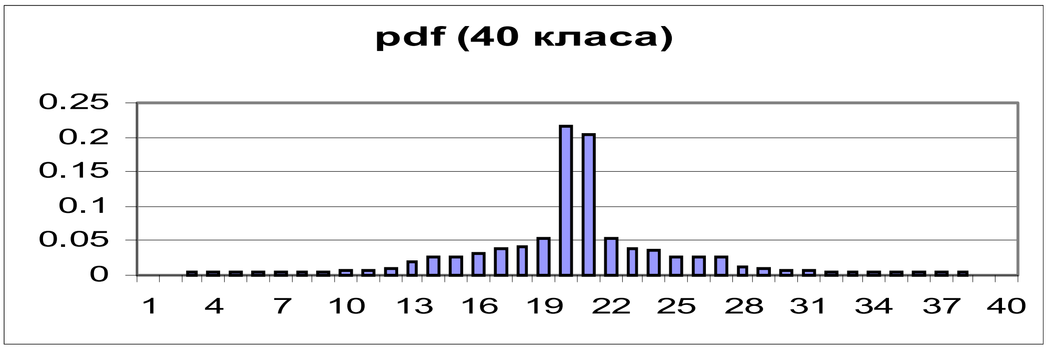

Figure 2. PDF for one-dimensional time series with vector quantization for 40 classes

On Figure 3 is given the probability density function, when one-dimensional time series is divided into 20 classes and there is 5% smoothing (for Figure 1 - down – right).

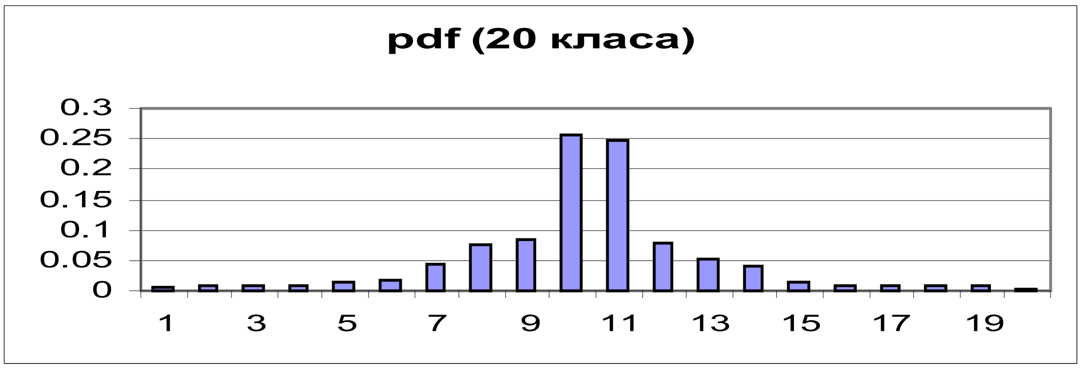

Figure 3. PDF for one-dimensional time series with vector quantization for 20 classes

DETERMINIG OF PDF FOR TWO-DIMENSIONAL VECTOR SERIES

Consider two-dimensional series, where, for example are taken values from the first second from the same accelerogram as previous section, where are about 500 values, depicted on axis North-South with acceleration values a_x and East-West with acceleration values a_Y. The input space is divided into 25 classes (5 on axis *X* and 5 on axis *Y*). Vector quantization is according to Algorithm 4. Received results on MatLab 6.5 are shown on Figure 4, where are depicted received centers of target classes after learning and training.

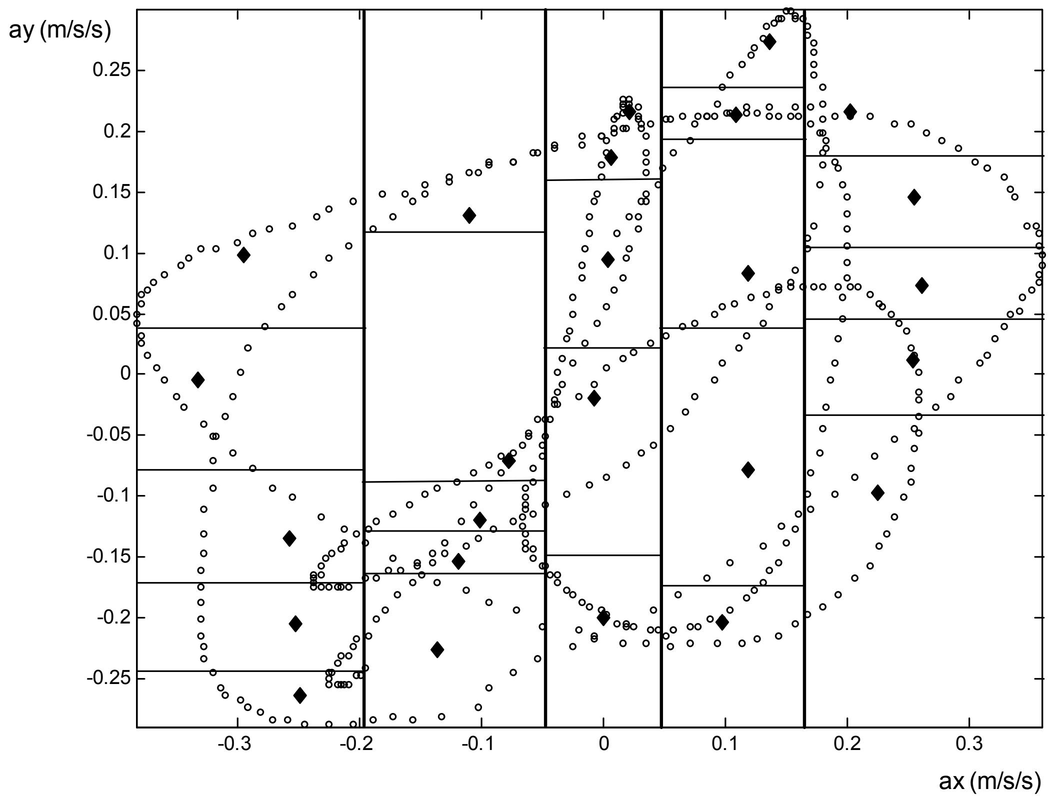

Figure 4. PDF for one-dimensional time series with vector quantization for 20 classes

CONCLUSIONS

In this work are suggested algorithms for one-dimensional and two-dimensional vector quantization for two-layered neural network. Kohonen learning rule is discussed and implemented for different number of target values in each class and with equally or not equally classes width. Numerical examples for determining PDF are shown.

The work is a part of the international research project PST.CLG.979333 with financial support of NATO scientific programs.

REFERENCES

[1] Kohonen, T.: Self-Organizing Maps. Second Edition, Springer Verlag, Berlin 1997.

[2] Radeva, S., Scherer, R., Radev, D., Yakov, V. Real-time estimation of strong motion seismic waves, *Acta Geodaetica et Geophysica*, Vol. 39 (2-3), 2004, pp. 297-308.

ABOUT THE AUTHORS

Assoc. Prof. Svetla Radeva, PhD. Department of Computer Aided Engineering, University of Architecture Civil Engineering and Geodesy, Sofia, Phone:+359 2 954 88 67, E-mail: svetla_fce@abv.bg

Assoc. Prof. Dimitar Radev, PhD, Eng. Department of Communication Technique and Technologies, University of Rousse, Phone: +359 82 888 683 E-mail: dradev@abv.bg

Eng. Vladimir Yakov, PhD student, Department of Computer Aided Engineering, University of Architecture Civil Engineering and Geodesy, Sofia, E-mail: vykv@yahoo.com