Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# On building local models for inverse system identification with vector quantization algorithms

Luís Gustavo M. Souza, Guilherme A. Barreto *

*Federal University of Ceará, Department of Teleinformatics Engineering, Av. Mister Hull, S/N - Center of Technology - Campus of Pici, CP 6005, CEP 60455-760, Fortaleza, Ceará, Brazil*

A R T I C L E   I N F O

Available online 6 March 2010

*Keywords:*
System identification
Vector quantization
Inverse modeling
Local models
Residual analysis
Hypothesis testing

A B S T R A C T

In this paper we provide a comprehensive performance evaluation of vector quantization (VQ) algorithms as building blocks for designing local models for inverse system identification. We describe how VQ algorithms can be used for learning compact representations of the task of interest from available input–output time series data and how this representation can be used to build local maps that approximates the global inverse model of the system. The performances of the resulting local models are compared to the standard global (multilayer perceptron) MLP-based model in the task of inverse modeling of four well-known single input–single output (SISO) systems. The obtained results show that VQ-based local models perform better than the MLP in all the studied tasks.

## 1. Introduction

System identification is concerned with the development of a regression model that describes the behavior of a dynamic system from measurements of its inputs and outputs. Knowing a model that describes the diversity of behaviors that a dynamic system can reveal, specially the nonlinear ones, is essential not only for theoretic or applied research fields, but also for the process or control engineer who is interested in understanding better the dynamics of the system he/she is dealing with. As an ultimate goal, the resulting model should imitate the actual system as faithfully as possible in order to be used for several additional purposes, such as predictive control or fault detection.

System identification techniques can be classified into two broad categories: global and local models. Global approaches adopt a single model to represent the input–output behavior of the system. In the context of the current paper, a global model comes in the form of a single neural network model, such as the multilayer perceptron (MLP) [34]. Global models represent the mainstream in applications of nonlinear system identification and control [35,36].

Local approaches utilize instead multiple models to represent the input–output behavior of the system [25]. In few words, the input space is usually divided into smaller (localized) regions, each one being associated with a simpler model. To estimate the system output at a given time, a single model is chosen from the pool of available local models according to some criteria defined on the current input data. In a nutshell, local models perform local linearization and their structure is quite similar to the Takagi–Sugeno fuzzy models [44].

If the local models are formulated as linear autoregressive models with exogenous inputs (ARX, for short), then we have a local ARX modeling approach, widely used within the adaptive control community [38]. This approach is closely related with the piecewise linear function approximation method, whose underlying idea is to specify a set of hyperplanes (that is, adaptive filters), each one locally used, in order to approximate the nonlinear surface that defines the input–output mapping of interest.

Obviously, the main advantage of the local modeling approach over the global one relies on the fact that complex (e.g. nonlinear) dynamics of the input–output mapping can be represented by simpler linear mappings. Another advantage is interpretability. Since local models are used, one can easily associate IF–THEN rules with them in order to describe the current status of the system. A clear disadvantage of local models is that the user has to define beforehand how many local models are going to be used.

In the neural network literature, local modeling techniques have been implemented through the use of the self-organizing map (SOM) [3,12,6,11,4,41]. The SOM [24] is an unsupervised competitive learning algorithm which has been commonly applied to vector quantization and data visualization tasks. The results reported on those studies are rather appealing, indicating that SOM-based local models can be feasible alternatives to global models based on supervised neural network architectures, such as the MLP and RBF (*radial basis function*). However, many questions still remain unanswered.

For example, concerning the VQ algorithm used to build the local models, what is the effect on the performance in case of using other VQ algorithm than the SOM? There are many VQ

* Corresponding author. Tel.: +55 85 3366 9467; fax: +55 85 3366 9468.
*E-mail addresses:* luisgustavo@deti.ufc.br (L.G. Souza), guilherme@deti.ufc.br (G.A. Barreto).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.10.021

algorithms available in the machine learning community, such as the *K*-means [30], standard winner-take-all (WTA) competitive learning [42], frequency sensitive competitive learning (FSCL) [1] and the fuzzy competitive learning (FCL) [13], to mention just a few of them. However, none of them so far have been used for the purpose of building local models for system identification. Would they perform better than the SOM in this regard? This is an important issue, since some of these VQ algorithms are computationally lighter than the SOM (e.g. *K*-means and FSCL).

Another important issue is concerned with the identification of inverse mappings through VQ-based local models. As far as we are aware, the VQ-based local modeling approach has not been applied to such task yet. The aforementioned works report results on the identification of forward models of the dynamic system of interest. This can be partly explained by the fact that inverse modeling is an ill-posed problem, subject to several difficulties that are hard to tackle, such as non-minimum phase systems and multiple solutions. Later in this paper we elaborate more on the issue of inverse modeling and its application.

And last but not least, an important issue is concerned with the evaluation of the performance of local models themselves. The vast majority of previous works on local modeling for system identification do not provide in-depth statistical analysis of the models' results. Usually, only MSE values are provided, but much more are required, such as the residual analysis and hypothesis testing.

In this paper we initially introduce strategies for building local inverse models using the SOM network. Then, we evaluate the performance of local inverse models built using the SOM and several other VQ algorithms on four different input–output datasets. The performances of these local models are also compared with those resulting from global MLP-based inverse models. Finally, a comprehensive analysis of the residues of all models are carried out in order to test how different they are from the statistical viewpoint.

The remainder of the paper is organized as follows. In Section 2, we discuss a bit further the issue of inverse modeling and its use in control systems. In Section 3, the SOM network and its learning process are briefly described. In Section 4, three SOM-based local modeling approaches are introduced. Simulations and performance are presented in Section 5. The paper is concluded in Section 6.

## 2. The inverse system identification task

There are many real-world dynamic systems which need to be monitored and controlled on the basis of readings obtained, for instance, from distributed sensors in an industrial plant [33,14]. In this scenario, decisions must be taken constantly in order to maintain the behavior of the system within appropriate limits. Common strategies for controlling the behavior of dynamic systems require forward and/or inverse models of the system. The Internal Model Control [22] architecture uses, for example, both the forward and inverse models for controlling a given system.

Inverse models should provide a faithful representation of the inverse mapping within the operational regions of interest. Inverse modeling is a signal processing task usually considered much more difficult to deal with than forward modeling, since multiple solutions may exist. Thus, inverse modeling belongs to the class of ill-posed problems.

There are several ways to carry out the inverse identification process. The most common technique used for this purpose is known as the generalized inverse learning method [20,21]. According to this method, the model is fed with the current desired output (or reference value) together with past inputs and past outputs to predict the current input (or control action). Fig. 1 illustrates the generalized inverse learning method. The neural network block denotes an adaptive global/local model that tries to learn the inverse mapping of the system. The desired output value, $u(t)$, corresponds to the required set point or reference signal and $\hat{u}(t)$ corresponds to the predicted output.

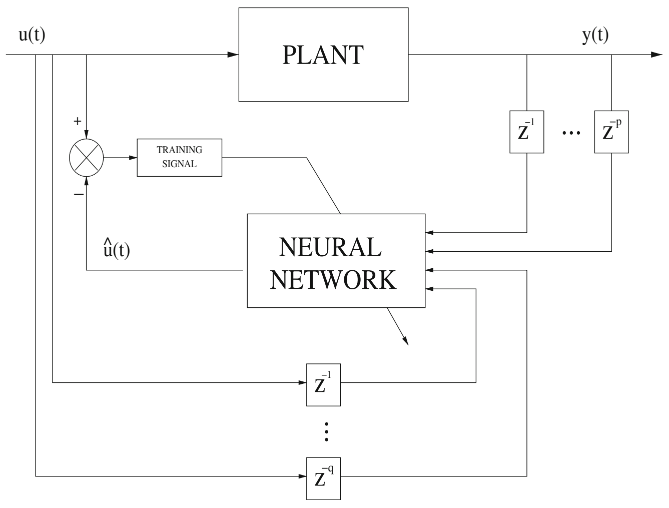

**Fig. 1.** Generalized method for the training of inverse neural networks for control.

In this paper, we evaluate the proposed VQ-based local modeling approaches on providing faithful inverse mappings of four benchmarking input–output systems. For this purpose we adopt the generalized inverse learning method strategy.

## 3. The self-organizing map

The *self-organizing map* (SOM) is a well-known competitive learning algorithm. The SOM learns from examples a mapping (projection) from a high-dimensional continuous input space $\mathcal{X}$ onto a low-dimensional discrete space (lattice) $\mathcal{A}$ of $N$ neurons which are arranged in fixed topological forms, e.g., as a rectangular two-dimensional array. The map $i^*(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{A}$, defined by the weight matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q)$, $\mathbf{w}_i \in \mathbb{R}^p \subset \mathcal{X}$, assigns to each input vector $\mathbf{x}(t) \in \mathbb{R}^p \subset \mathcal{X}$ a winning neuron $i^*(t) \in \mathcal{A}$, determined by

$$i^*(t) = \arg\min_{\forall i} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean distance and $t$ symbolizes a discrete time step associated with the iterations of the algorithm.

The weight vector of the current winning neuron as well as the weight vectors of its neighboring neurons are simultaneously adjusted according to the following learning rule:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)], \tag{2}$$

where $0 < \alpha(t) < 1$ is the learning rate and $h(i^*, i; t)$ is a weighting function which limits the neighborhood of the winning neuron. A usual choice for $h(i^*, i; t)$ is given by the Gaussian function:

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right), \tag{3}$$

where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$ are, respectively, the coordinates of the neurons $i$ and $i^*$ in the output array, and $\sigma(t) > 0$ defines the radius of the neighborhood function at time $t$. The variables $\alpha(t)$ and $\sigma(t)$ should both decay with time to guarantee convergence of the