Neural Comput & Applic (2009) 18:345–357
DOI 10.1007/s00521-008-0185-3

ORIGINAL ARTICLE

Prediction of indoor temperature and relative humidity using neural network models: model comparison

Tao Lu · Martti Viljanen

Received: 7 June 2007 / Accepted: 4 April 2008 / Published online: 6 May 2008

Abstract The use of neural networks grows great popularity in various building applications such as prediction of indoor temperature, heating load and ventilation rate. But few papers detail indoor relative humidity prediction which is an important indicator of indoor air quality, service life and energy efficiency of buildings. In this paper, the design of indoor temperature and relative humidity predictive neural networks in our test house was developed. The test house presented complicated physical features which are difficult to simulate with physical models. The work presented in this paper aimed to show the suitability of neural networks to perform predictions. Nonlinear AutoRegressive with eXternal input (NNARX) model and genetic algorithm were employed to construct networks and were detailed. The comparison between the two methods was also made. Applicability of some important mathematical validation criteria to practical reality was examined. Satisfactory results with correlation coefficients 0.998 and 0.997 for indoor temperature and relative humidity were obtained in the testing stage.

Keywords Neural networks · Indoor relative humidity prediction · Indoor temperature prediction · NNARX model · Genetic algorithm · Model validation

T. Lu (✉) · M. Viljanen
Laboratory of Structural Engineering and Building Physics, Department of Civil and Environmental Engineering, Helsinki University of Technology, P.O. Box 2100, 02015 Espoo, Finland
e-mail: tlu@cc.hut.fi

1 Introduction

Moisture problem is one of the most serious factors in building and housing industry. Over the last decade, moisture failures in building systems have reached billions of Euros in damages in Europe, many of which involved the deterioration of sheathing panels and energy efficiency. Additionally, excess moisture in envelopes can lead to the presence of molds which results in poor indoor air and causes health problems of the inhabitants [1–2]. Thus indoor moisture prediction becomes the part of import work prior to indoor air quality control.

Over the decades, many researchers have devoted to such modelling topics. There are many models available. In our laboratory for instance, an accurate numerical model of coupled heat and moisture transfer in buildings has been developed [3]. More detailed and complicated models are Navier–Stokes equations which describe the flow of fluids for airflow, temperature and contaminant distributions. A computational fluid dynamics (CFD) technique is employed to handle these equations. Teodosiu et al. [4] employed a CFD technique and a modified k–ε turbulence model to predict indoor air moisture and its transport in a mechanically ventilated test room to estimate the level of thermal comfort. Experimental–numerical comparisons with regard to thermal comfort indices were also provided [4]. The model is very useful in studies dealing with thermal comfort predictions where an exact distribution of indoor air moisture is required.

These numerical methods, called physical models, can simulate inside climate environment and airflow distribution even before building is constructed. This is one of their advantages. However, these numerical methods typically require a lot of computation and lead to time-consuming simulations. Take CFD models as an example. Although

CFD models can give highly detailed results, the implied accuracy of the results is defined by the assumptions inherent in the model setup, thus, there is the potential of a very costly and refined computation. For a medium-size building, it may take days to complete indoor temperature simulation in a modern personal computer (PC) [5]. Therefore, most of the CFD models are limited to steady state calculations as the model developed by Teodosiu et al. [4].

Most importantly, a general drawback of these models is that the output of the model is only as accurate as input physical data, for example airflow rate was needed in the CFD model developed by Teodosiu et al. [4]. Presently there are many buildings whose input physical data are poorly defined, which creates ambiguity or uncertainty in predicting and interpreting the output. Physical models fail to account for these complicated cases. For example, in a central ventilation control room where inside climate gets great impacts by ventilation machine activities, it is almost impossible for physical models to simulate indoor climate because too many unknown factors are involved. On the other hand, a black-box model, such as neural networks, can deal such extreme cases without much difficulty. Unlike physical models, neural networks entirely depend on experimental data which can be made adaptive and offer a much faster computation. Compared to physical models, a neural network takes just a few minutes to finish indoor climate forecast for a medium-size building [5]. Physical and neural network models are complementary.

However, indoor humidity prediction with neural network models is lacking in literature due to its more complicated mechanism involved which depends on thermal behaviours or temperature prediction. Sigumonrong et al. [6] used historical data to predict indoor temperature and relative humidity, yet their main focus was indoor relative humidity maintenance rather than prediction. No details were provided on input variable identification and prediction results. Similar work was done by Zhang et al. [7]. Concerning indoor temperature prediction, some literatures exist. Ferreira et al. [8] adopted RBF (radial basis function) neural network model to predict indoor temperature for a green house. Using RBF, Ruano et al. [5] predicted indoor temperature for a school building where a genetic algorithm was also employed for searching optimal structure for neural networks. Thomas et al. [9] investigated indoor temperatures for two buildings using feedforward neural networks.

Despite the efforts these works made, there are still modelling issues that have not been touched. First, for both indoor temperature and relative humidity, the actual prediction situations involved in these works are not very complicated. Impact factors can be well identified and unknown factors have little impact on predictions. Secondly, the prediction of indoor relative humidity using neural networks is not detailed. No detailed information is provided on how to identify input variables and how to search optimal structures for indoor relative humidity. Thirdly, most papers on indoor temperature predictions pay great attention on training stage, such as optimal structure search and input variable identification, and give less attention on validation stage. Criteria like MSE (the mean of square errors), MAE (the mean of square errors) and SSE (the sum of square errors) for accuracy test in validation stage are commonly adopted. However, it is insufficient for these criteria to address problems, such as whether the network is uncertain to a particular input as well as overfitting and underfitting problems.

In this paper, we address these questions. We focus on the design of neural networks for indoor temperature and relative humidity predictions for one of our test houses, a central ventilation control room. The choice of the test house stemmed from the following facts: as introduced before, moisture is one of the primary causes of damage observed in building structures, increasing the importance of the development of research with the aim of finding regulations of controlling moisture in buildings subject to outdoor climates. Using ventilation to control moisture is an obvious way and it can improve indoor air quality also. For this purpose, we need to pursue more flexible simulation models than physical models to account for the possible complications that can occur in association with ventilation and heating systems. We selected somewhat more complicated test house for studying neural network models. The methodologies proposed in this paper work perfect well for general residential and commercial buildings. The objectives of this study are:

- to examine prediction suitability of neural networks in terms of indoor temperature and relative humidity in a more complicated case where physical models are unable or very difficult to apply. Measurement data were taken from our test house in which indoor temperature and relative humidity were very much affected by ventilation machines. Innegligible amount of heat was generated from these ventilation machines. The impact factors are difficult to identify and to quantify such as ventilation and heating rates. Therefore, we chose neural network models.
- to investigate some important validation criteria and related techniques. Average generalization error, prediction interval and *k*-step ahead prediction are three validation criteria which are used to detect network uncertainty, overfitting and underfitting problems. These criteria have been widely discussed and implementation techniques have been well proved mathematically [10–12]. However, in practice they

are applied rarely, particularly in building application field. We aim to examine two important techniques: *k*-step ahead prediction [10] and delta method [11] for prediction interval. Other techniques are also discussed;
- to apply two different approaches, NNARX and genetic algorithm, to search optimal network structures for indoor temperature and relative humidity predictions. Both techniques have been studied in indoor temperature prediction, but not in indoor relative humidity in literature. We employ the two methods for both indoor temperature and relative humidity prediction and present the comparison results.

The layout of the paper is as follows: Sect. 2 describes experiments. Sect. 3 introduces modelling techniques including NNARX and genetic algorithm. Neural network structure and validation criteria including residual analysis, average generalization error, prediction interval and *k*-step ahead prediction are also presented in Sect. 3. Section 4 reports model results and discussion. Prediction intervals for all developed networks are particularly mentioned in Sect. 4. Section 5 addresses conclusion and future work.

2 Experiments

The dataset was obtained from the weather station located inside our test house, a central ventilation control room (Fig. 1). The ventilation room is on the top of the department building with outdoor temperature/humidity sensors mounted on the roof. The room has three big and one small ventilation machines which are responsible for ventilating the half of the building. These machines constantly generate innegligible amount of heat which is hard to measure. The dimension of the test house is 147.8 $m^2 \times 3$ m. Ventilation ducts are distributed all over the room. It is easy to see that such test house present complicated indoor characteristics that are difficult to describe physically.

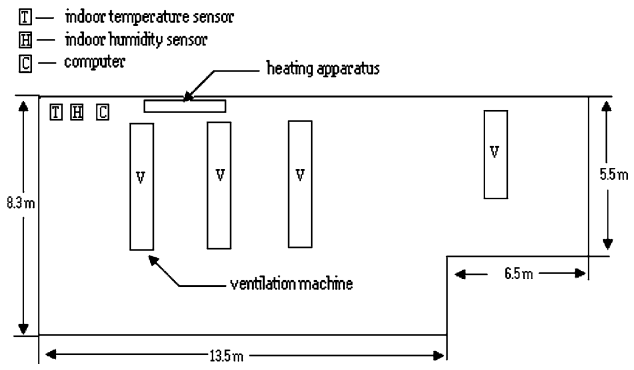

Fig. 1 Room layout

The experiment was carried out from January 2007 for 30 days. All the variables, temperatures and relative humidity indoor and outdoor, were measured within a 15 min interval. The total sample size is 2,930. However, no outdoor relative humidity was available for outdoor temperature below −10°C which gave the actual 2,667 samples for indoor relative humidity. No information was obtained for machine generated heating power, electricity power and ventilation rate due to their complicated features which were difficult to quantify. The first 1,543 and 1,690 samples were chosen as training data for indoor relative humidity and temperature, respectively.

The following variables are used for indoor temperature and relative humidity predictions:

t	Time (time interval is 15 min).
$T_o(t)$	Outdoor temperature, °C.
$T_i(t)$	Indoor temperature, °C.
$RH_o(t)$	Outdoor relative humidity, %.
$RH_i(t)$	Indoor relative humidity, %.

Figure 2 shows all records for the above variables.

3 Models

Nonlinear autoregressive with external input model and genetic algorithm were used to construct networks for our predictions.

3.1 NNARX

In engineering field, the popular approach is to model the nonlinear system by using existing nonlinear autoregressive moving-average with exogenous inputs type of model [13]. This kind of models may include errors as inputs (NARMAX) or not (NARX) [10]. Here, we just discuss the nonlinear autoregressive with exogenous inputs (NARX). NARX has the following form:

$$y(t) = f[y(t-1), \ldots, y(t-n_a), u(t-k), \ldots, u(t-k-n_b+1)] + e(t) \quad (1)$$

where y and u represent observed outputs and inputs respectively, n_a and n_b the orders and k the delay which can be identified by minimizing Akaike's Information Criterion or Rissanen's minimum description length [10].

Nonlinear autoregressive with exogenous inputs model has several advantages. In addition to its simplicity, the major advantage is its natural connection to conventional identification techniques. NNARX is an extended NARX neural network model containing weights and bias **w** as

$$y(t) = f[\varphi(t), \mathbf{w}] + e(t) \quad (2)$$

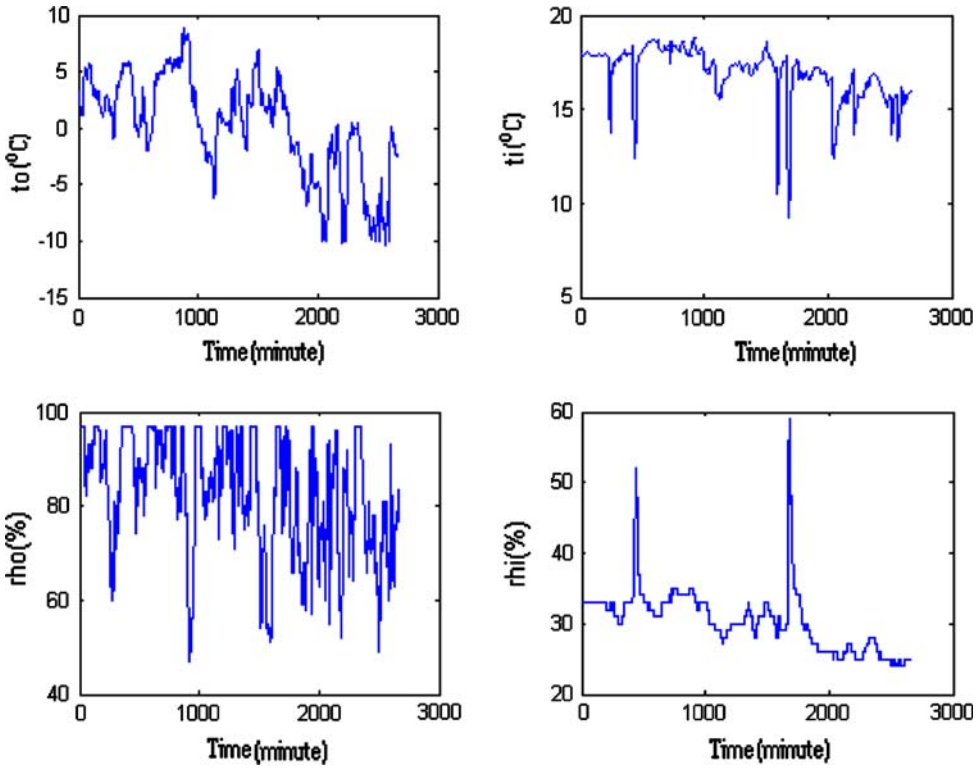

Fig. 2 Outdoor temperature (*top left*), indoor temperature (*top right*), outdoor relative humidity (*bottom left*) and indoor relative humidity (*bottom right*) (all time intervals are 15 min)

where

$$\varphi(t) = [y(t-1)\ldots y(t-n_a)\ u(t-k)\ldots u(t-k-n_b+1)]^T \tag{3}$$

with the prediction form

$$\hat{y}(t|\mathbf{w}) = f[\varphi(t), \mathbf{w}].$$

In terms of indoor temperature and relative humidity, input variables could be outdoor temperature (T_o), outdoor relative humidity (RH_o), heating power, ventilation rate and previous indoor temperature (T_i) and relative humidity (RH_i) in physical insights. However, as data on heating power and ventilation rate were not available, we took the following model structures:

$$\hat{T}_i(t|\mathbf{w}_{ti}) = g_{ti}(\varphi_{ti}(t), \mathbf{w}_{ti}) \tag{4}$$

$$\widehat{RH}_i(t|\mathbf{w}_{rhi}) = g_{rhi}(\varphi_{rhi}(t), \mathbf{w}_{rhi}) \tag{5}$$

where

$$\varphi_{ti} = [T_i(t-1)\ldots T_i(t-n_a)T_o(t-k)\ldots T_o(t-k-n_b+1)]^T \tag{6}$$

$$\begin{aligned}\varphi_{rhi} =& [RH_i(t-1)\ldots RH_i(t-n_c)RH_o(t-k_d)\ldots \\ & RH_o(t-k_d-n_d+1)T_i(t-k_e)\ldots T_i(t-k_e-n_e+1) \\ & T_o(t-k_f)\ldots T_o(t-k_f-n_f+1)]^T.\end{aligned} \tag{7}$$

Here n and k with subscripts are orders and delays which can be determined by minimizing the same criteria as NARX model. In our study, all orders and delays were restricted to [1, 5]. For each combination, optimal values (i.e. delays and orders) were calculated using Matlab System Identification Toolbox.

3.2 Neural network structure

The selection of layer number for a neural network is the first and crucial step for neural network models, and it will make great impact on the following implementation, training, validation as well as statistical analysis of the neural networks. In our work, three-layer feedforward neural network was chosen as our fundamental structure for all neural networks. This decision is based on the following facts:

1. The model we wish to adopt has to meet the common model selection criteria of principle of parsimony that no more parameters than necessary should be used.
2. It has been proved that a three-layer feedforward neural network can approximate any function as long as a sufficient number of hidden neurons are provided [14]. The broad range of model works, as well as the results performed in this study, indicates three-layer feedforward neural network can adequately model the data [15].
3. More sophisticated network models, for example four or more layers of feedforward neural networks, can lead to more complicated implementation, training,

validation and statistical analysis, and by far theoretically and practically it is not quite sure whether the accuracy can be really improved when using a more sophisticated network. So, it is customary to apply three-layer neural networks with one input layer, one hidden layer and one output layer in practice, particularly in engineering applications.

The structure of a three-layer feedforward neural network with two inputs, two hidden units and one output is illustrated as follows:

W_{ij} and w_{ij} are weights, b_{ij} and B_{ij} are bias. The mathematical formula for Fig. 3 is:

$$\hat{y}(t) = F_1\left[\sum_{j=1}^{n_h} W_{1j} f_j\left(\sum_{i=1}^{n_u} w_{ji}u_i + b_{j0}\right) + B_{10}\right] \tag{8}$$

n_h is the number of hidden units and n_u is the number of input variables. In our study, f_i is often taken as hyperbolic tangent function and F_1 as linear function. So, Eq. (8) can be rewritten as:

$$\hat{y}(t) = \sum_{j=1}^{n_h} W_{1j} \tanh\left(\sum_{i=1}^{n_u} w_{ji}u_i + b_{j0}\right) + B_{10} \tag{9}$$

The goal of neural networks is to minimize error functions between predictions and measurements by adjusting weights and biases. After input variables are identified, the selection of the size of hidden units becomes crucial for the performance of a neural network. There is no exact rule about how to define the size of hidden units, but still some papers [16] suggest the maximum number of elements in the hidden layer to be twice the input layer dimension plus one. Here, we set the number of hidden units as the same as the number of input variables. But, as for a fully connected neural network, overfitting becomes another problem. So-called overfitting means network performs well in training stage, but bad in testing. The cause of overfitting is complex enough to memorize data in training stage. Normally overfitting can be solved by minimizing network's complexity, namely eliminating a number of unnecessary weights in the sense that the network is either not too complex to cause overfitting or not too simple to learn training data, in other words, try to find an optimal network topology. There are two techniques to determine the optimal network topology. One is called optimal brain surgeon (OBS), another one is called optimal brain damage (OBD). In principle, both cases start out initially with relatively large networks and then successively prune network branches (weights) of one at a time until the optimal architecture has been found [10]. The difference between these two is that OBD only prunes weights, but OBS also adjusts the remaining weights. In this study, we accepted OBS as our pruning method and the networks were trained using fast training algorithm-Levenberg–Marquardt [17–18] in 300 epochs. Matlab based Neural Network Based System Identification Toolbox [19] and Neural Network Toolbox were used as training tools.

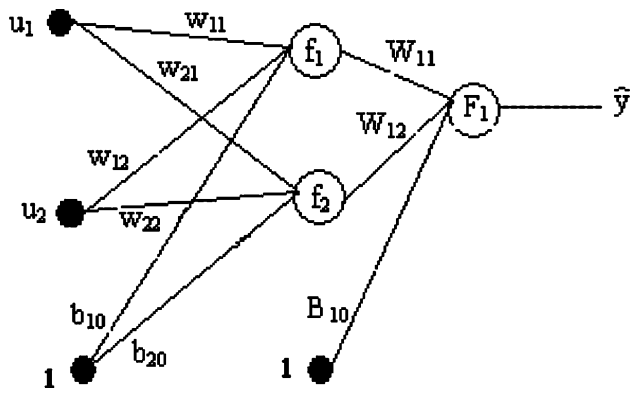

Fig. 3 Three-layer feedforward neural network with two inputs, two hidden units and one output

3.3 Model validation criteria

The mean of the sum of square errors (MSE) and the mean of absolute errors (MAE) are accepted as two key criteria to evaluate the fit to the identification data.

$$\text{MSE} = 1/N\sum_{t=1}^{N}[\hat{y}(t) - y(t)]^2 \tag{10}$$

$$\text{MAE} = 1/N\sum_{t=1}^{N}|\hat{y}(t) - y(t)| \tag{11}$$

where $\hat{y}(t)$ is predicted output and $y(t)$ is measured one.

Ideally, the validation should be performed in accordance with the intended use of the model [10]. In our case, the intention was to predict indoor temperature and relative humidity at least one-step ahead and the completed networks were able to estimate prediction intervals for unseen data, not just point estimation, in other words, networks are capable of detecting uncertainties for unseen data. So the above two criteria are not enough to validate these performances for a network in terms of MSE and MAE. Therefore, other validation methods should be also considered.

3.3.1 Average generalization error

The estimate of the average generalization error is on purpose to validate the generalization ability of neural networks, but the main application is for model selection. There are two prime techniques for estimating the average generalization error: leave-one-out cross-validation and bootstrap. Leave-one-out cross-validation, also called LOO (Leave-One-Out) for short, is commonly used in linear

regression and it requires a great amount of computation in neural network applications [10]. For an N size training data, LOO needs N times of network retraining. For a large number of training data, it is almost impossible in practice. However, LOO's alternative version, linear-unlearning-leave-one-out (LULOO) [20], does not require any retraining and offers a much faster calculation. The payoff is that LULOO is in general less accurate and reliable. In this paper, we examined LULOO for estimating the average generalization error, but results were not taken into account. Bootstrap is by far the most accurate method for estimating the average generalization error [21]. But, despite that bootstrap demands less computation than LOO, it still needs a considerable amount of calculation.

3.3.2 Prediction interval

Interval estimate expands on point estimate by incorporating the uncertainty of the point estimate which is more preferred. In addition to prediction purpose, such prediction interval can also show the reliability of a specific input to the network. There are two major methods for estimating prediction interval: delta method and bootstrap [11]. Bootstrap is more accurate. However, it has a limitation on heavy computation. For a network model, bootstrap requires hundreds of times of retraining. Delta method instead gives a fast calculation but less accurate and requires Hessian matrix calculation. The Hessian matrix calculation often causes failure in practice. An alternatively more practical method was derived from delta method to estimate prediction interval [22]. The method is described in the following.

For a given set of N pairs data $\{(y_n, x_n)\}_{n=1}^{N}$, suppose the neural network

$$\hat{y}(\mathbf{X}|\mathbf{w}) = f(\mathbf{X}, \mathbf{w}) \tag{12}$$

where $\hat{y}(\mathbf{X}|\mathbf{w})$ is predicted value under the weight $\mathbf{w}$. Let $\mathbf{J}$ be a matrix whose ijth entry is $\partial f(\mathbf{x}_i)/\partial w_j$ evaluated at the true parameter vector $\mathbf{w}^*$ (supposing the true model exists) and $\mathbf{x}_i$ is ith input (ith row of $\mathbf{X}$) and $\mathbf{g}_0$ is a vector whose ith entry is $\partial f(\mathbf{x}_0)/\partial w_i$, evaluated at the true parameter $\mathbf{w}^*$ (in practice $\mathbf{w}^*$ can be replaced by estimated weight, $\hat{\mathbf{w}}$) and $\mathbf{x}_0$ is a specific input. Then the prediction interval (half width) is

$$t_{n-p} s \sqrt{1 + \mathbf{g}_0'(\mathbf{J}'\mathbf{J})^{-1}\mathbf{g}_0} \tag{13}$$

where t_{n-p} is t distribution with n-p degrees of freedom, $s = \sqrt{\mathrm{RSS}/(n-p)}$ for $\mathrm{RSS} = \sum (y_i - \hat{y}_i)^2$ and p the number of parameters including ^{i}weights and biases. If weight decay is accounted, then Eq. (13) becomes

$$t_{n-p^*} s \sqrt{1 + \mathbf{g}_0'(\mathbf{J}'\mathbf{J} + \alpha\mathbf{I})^{-1}(\mathbf{J}'\mathbf{J})(\mathbf{J}'\mathbf{J} + \alpha\mathbf{I})^{-1}\mathbf{g}_0} \tag{14}$$

where α is the decay, $s = \sqrt{\mathrm{RSS} + \alpha \sum_i^p w_i^2/(n-p*)}$ and $p^* = tr(2\mathbf{J}(\mathbf{J}'\mathbf{J}+\alpha\mathbf{I})^{-1}\mathbf{J}' - (\mathbf{J}(\mathbf{J}'\mathbf{J}+\alpha\mathbf{I})^{-1}\mathbf{J}')^2)$.

Though results from Eqs. (13, 14) are less accurate than those obtained from *bootstrap* especially when sample size is small, no training is needed. In practice, $(\mathbf{J}'\mathbf{J})^{-1}$ is estimated using inverse Hessian $\mathbf{H}^{-1}$. Here, the elements of Hessian matrix $\mathbf{H}$ are the second-order partial derivatives $H_{i,j} = \frac{\partial^2 \mathrm{Err}(\mathbf{w})}{\partial w_i \partial w_j}$ evaluated at $\mathbf{w} = \hat{\mathbf{w}}$, where $\mathrm{Err}(\mathbf{w})$ is the error function of $1/2\sum_i (y_i - \hat{y}_i)^2$ in this study. Bishop presented a detail algorithm on the calculation of inverse Hessian whose convergence is guaranteed despite that long computation time may be needed [23]. In our work, we used Eq. (13) to estimate prediction intervals for models generated by genetic algorithm and Eq. (14) for models from NNARX. All the prediction intervals have 95% confidence.

3.3.3 K-step prediction

The k-step ahead prediction is able to reveal whether important information is captured by the model or not. It is often taken as an auxiliary tool to detect underfitting and overfitting problems. The k-step ahead prediction is based on one-step prediction. In NNARX model, the k-step ahead prediction has the following form for example (see [10] for detail):

$$\hat{y}(t+k) \triangleq \hat{y}(t+k/t, \hat{\theta}) = \hat{g}\left[\hat{\varphi}(t+k), \hat{\theta}\right] \tag{15}$$

where

$$\begin{aligned}\hat{\varphi}^T(t+k) = [&\hat{y}(t+k)\ldots\hat{y}(t+k-\min(k,n)+1)y(t)\ldots \\ &y(t-\max(n-k,0))u(t-d+k)\ldots \\ &u(t-d-m+k)]\end{aligned}$$

A four-step ahead prediction was adopted in our model.

3.4 Genetic algorithm

The genetic algorithm is a method for solving optimization problems originally inspired by biological evolution. The algorithm encodes a potential solution to a specific problem to a chromosome-like structure and applies recombination operators to these structures in order to preserve critical information. The genetic algorithm starts with an initial population and then selects parents to produce next generation using specific rules. Three main rules are

Selection: Select individuals that contribute directly to the population at the next generation;

Crossover: Combine two parents to form children for the next generation;

Mutation: Make random change to parents to form children.

Genetic algorithm has been widely used in neural networks, of which the following three areas are the main applications:

- set weights for a fixed structure;
- learn network topologies;
- select training data and interpret the output behaviour of neural networks.

A whole view on genetic algorithm in neural networks can be found in [24]. In this study, genetic algorithm was employed to determine input variables and the number of hidden units. Each combination of input variables is given as a binary representation while the number of hidden units as a ten-based number and each input ranges from one to five. For indoor temperature prediction, a chromosome-like code for each combination is

$$T_{\mathrm{i}}(t-1)T_{\mathrm{i}}(t-2)T_{\mathrm{i}}(t-3)T_{\mathrm{i}}(t-4)T_{\mathrm{i}}(t-5)T_{\mathrm{o}}(t-1)\ T_{\mathrm{o}}(t-2)T_{\mathrm{o}}(t-3)T_{\mathrm{o}}(t-4)T_{\mathrm{o}}(t-5)h_n \quad (17)$$

Similarly, for indoor relative humidity, we get

$$T_{\mathrm{i}}(t-1)T_{\mathrm{i}}(t-2)T_{\mathrm{i}}(t-3)T_{\mathrm{i}}(t-4)T_{\mathrm{i}}(t-5)T_{\mathrm{o}}(t-1)\ T_{\mathrm{o}}(t-2)T_{\mathrm{o}}(t-3)T_{\mathrm{o}}(t-4)T_{\mathrm{o}}(t-5)\mathrm{RH}_{\mathrm{i}}(t-1)\ \mathrm{RH}_{\mathrm{i}}(t-2)\mathrm{RH}_{\mathrm{i}}(t-3)\mathrm{RH}_{\mathrm{i}}(t-4)\mathrm{RH}_{\mathrm{i}}(t-5)\mathrm{RH}_{\mathrm{o}}(t-1)\ \mathrm{RH}_{\mathrm{o}}(t-2)\mathrm{RH}_{\mathrm{o}}(t-3)\mathrm{RH}_{\mathrm{o}}(t-4)\mathrm{RH}_{\mathrm{o}}(t-5)h_n \quad (18)$$

where h_{n} is the number of hidden units ranging from 2 to 20 for indoor temperature and 2 to 40 for indoor humidity. For example, the sequence 11000001114 presents $T_{\mathrm{i}}(t-1)$, $T_{\mathrm{i}}(\mathrm{t}-2)$, $T_{\mathrm{o}}(t-3)$, $T_{\mathrm{o}}(t-4)$ and $T_{\mathrm{o}}(t-5)$ and the number of hidden units is 4. MSE is set as an objective function. Hence the possible combinations are 19456 and 40900000 for indoor temperature and relative humidity, respectively. The fast algorithm Levenberg–Marquardt was used to train networks for both cases in 100 epochs. Mutation rate, crossover rate, migration rate were set to 0.05, 1 and 0.2 separately. Initial population and the number of generations were 50 and 100 for indoor temperature and 100 and 200 for indoor relative humidity, respectively.

4 Results and discussion

We first set $n_a = 3$, $n_b = 5$, $k = 1$ in Eq. (6) and $n_c = 2$, $n_d = 5$, $n_e = 3$, $n_f = 2$, $k_d = 1$, $k_e = 1$, $k_f = 2$ in Eq. (7) for NNARX model, which gave the following indoor temperature inputs

$$T_{\mathrm{i}}(t-1)T_{\mathrm{i}}(t-2)T_{\mathrm{i}}(t-3)\ T_{\mathrm{o}}(t-1)T_{\mathrm{o}}(t-2)\ T_{\mathrm{o}}(t-3)T_{\mathrm{o}}(t-4)T_{\mathrm{o}}(t-5) \quad (19)$$

and indoor relative humidity inputs

$$\mathrm{RH}_{\mathrm{i}}(t-1)\mathrm{RH}_{\mathrm{i}}(t-2)\mathrm{RH}_{\mathrm{o}}(t-1)\mathrm{RH}_{\mathrm{o}}(t-2)\mathrm{RH}_{\mathrm{o}}(t-3)\ \mathrm{RH}_{\mathrm{o}}(t-4)\mathrm{RH}_{\mathrm{o}}(t-5)T_{\mathrm{i}}(t-1)T_{\mathrm{i}}(t-2)T_{\mathrm{i}}(t-3)\ T_{\mathrm{o}}(t-2)T_{\mathrm{o}}(t-3) \quad (20)$$

Note that the number of hidden units was set as that of the input variables. After training, OBS was used to prune networks for the optimal structures of both indoor temperature and relative humidity predictions. Only $T_{\mathrm{i}}(t-1)$ and $T_{\mathrm{i}}(t-2)$ were finally identified as significance for indoor temperature prediction $T_{\mathrm{i}}(t)$. For indoor relative humidity $\mathrm{RH}_{\mathrm{i}}(t)$, the identified optimal terms were $\mathrm{RH}_{\mathrm{i}}(t-1)$, $T_{\mathrm{i}}(t-2)$, $T_{\mathrm{o}}(t-2)$ and $\mathrm{RH}_{\mathrm{o}}(t-1)$. MSE and MAE were obtained as 0.0062 and 0.0481 (testing) for indoor temperature while 0.2390 and 0.14 for indoor relative humidity. Figure 4 shows the comparison results.

For indoor relative humidity, we got five best models after running 200 generations in genetic algorithm. Similarly, two best models were obtained from 100 generations for indoor temperature. These results are displayed in Tables 1 and 2.

Tables 1 and 2 demonstrate that both temperature and relative humidity exhibit a time-lag effect. In the final best fit models, while a first order phase lag $T_{\mathrm{i}}(t-1)$ (approximately 15-min lag time) was observed for temperature, the

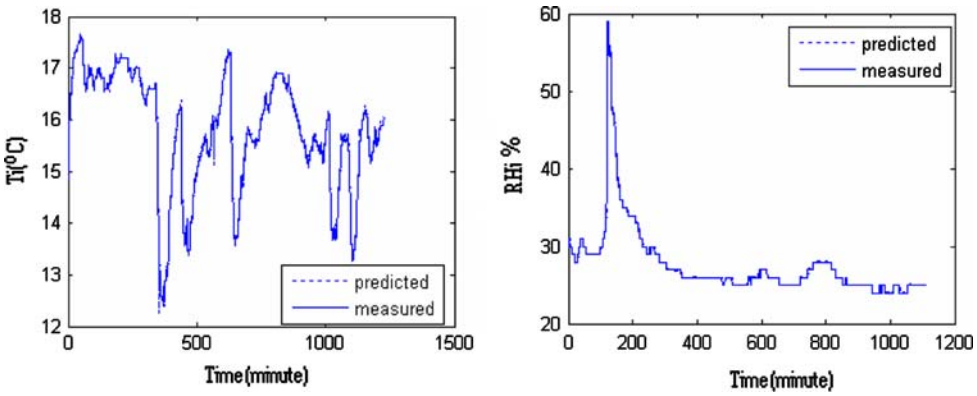

Fig. 4 The comparison between predicted indoor temperature and measured (*left*) and the comparison between predicted indoor relative humidity and measured (*right*) (NNARX) (all time intervals are 15 min)

Table 1 Results of five best models after 200 generations for indoor relative humidity prediction

Model	T_i	To	RH_i	RH_o	The number of hidden units	MSE
No 1			1, 2, 3, 5 (5 times)		6	0.2794
No 2			1, 2, 3 (2 times)		6	0.288
No 3		2	1, 2		2	0.294
No 4		2	1, 2, 3 (2 times)		6	0.295
No 5		2	1, 2, 3		2	0.3

"1, 2, 3, 5 (5 times)" means the combination of $RH_i(t-1)$, $RH_i(t-2)$, $RH_i(t-3)$, $RH_i(t-5)$ appears for five times in the final population

Table 2 Results of the best two models for indoor temperature prediction after 100 generations

Model	T_i	T_o	The number of hidden units	MSE
No 1	1, 3, 5 (4 times)		6	0.0104
No 2	1, 2 (3 times)		4	0.013

"1, 3, 5 (4 times)" means the combination of $T_i(t-1)$, $T_i(t-3)$, $T_i(t-5)$ appears for four times in the final population

order of phase lag for relative humidity was three with $RH_i(t-1)$, $RH_i(t-2)$ and $RH_i(t-3)$ (approximately 45-min lag time). The time lag is much longer for indoor relative humidity. Subsequently we assigned $T_i(t-1)$, $T_i(t-3)$ and $T_i(t-5)$ as inputs and set the number of hidden units as 6 for indoor temperature. For indoor relative humidity, $RH_i(t-1)$, $RH_i(t-2)$, $RH_i(t-3)$ and $RH_i(t-5)$ were chosen as inputs and the number of hidden units was set as 6. We retrained two networks with fast algorithm Levenberg–Marquardt in 200 epochs. Finally we obtained the models with improved performances as demonstrated in Table 3 and Fig. 5.

Tables 4, 5 show the comparisons of one-, two-, three- and four-step ahead indoor temperature and relative humidity prediction for NNARX and genetic algorithm.

We make the following observations on the results and discussion.

- Three-layer feedforward neural network works well in our work as we expected. All the four three-layer feedforward networks (NNARX and genetic models) possess satisfactory accuracies for predictions of indoor temperature and relative humidity.
- Indoor temperature predictions perform much better than those of indoor relative humidity. Mechanically, relative humidity is governed by nonlinear diffusion equation while temperature by linear diffusion equation. Indoor relative humidity is more difficult to predict.
- Both methods (i.e. NNARX and genetic algorithm) can provide four-step ahead indoor temperature predictions with great accuracies. However only with genetic algorithm we could obtain a satisfactory accuracy for a two-step ahead indoor relative humidity prediction because, after two-step ahead prediction, MAE of relative humidity prediction from both methods is less than their MSE, meaning that errors from some areas are getting much greater than 1 and as a result these areas are becoming unpredictable. These facts also indicate that indoor relative humidity may be influenced by more factors than indoor temperature especially in this study where some important information, such as heating power and ventilation rate, is unavailable, indoor relative humidity becomes much more difficult to predict.
- The mean of absolute errors of indoor relative humidity from NNARX method is less than its MSE. This implies that the error at some point might be very big and as a consequence the network could not capture the entire information for that point. Unfortunately, we cannot pinpoint the point with the biggest error in Fig. 4. However, we further conducted Matlab calculation and located the biggest error point (error 8.0749) where the measured value is 53 and the predicted value is only 44.9251. At this point, the model gave only 84.76% accuracy. On the other hand, the statistics shows the biggest error from genetic algorithm is 3.6697 which is produced by a predicted value 55.6697 with respect to the measured value 52. The accuracy can reach 92.94%. It should be noted that by only comparing the biggest predicted errors in validation stage cannot give a full picture on which model is superior. A further investigation must be done to show each model's possible error intervals. A full prediction interval analysis is presented in Fig. 6 and Table 6.
- Indoor temperature predictions from both methods perform better in testing stage than in training stage. This is an indication that obtained networks for indoor temperature are capable of generalizing.
- It seems that increasing the number of inputs does not improve prediction accuracy, as all the best four models preserve small number of inputs. This indicates that more inputs complicate the system which in turn is difficult for the network to extract information.

Table 3 Results from genetic algorithm

Model	Training performances	Testing performances
RH_i		
MSE	0.0446	0.1584
MAE	0.0624	0.1717
T_i		
MSE	0.023	0.0080
MAE	0.0608	0.0561

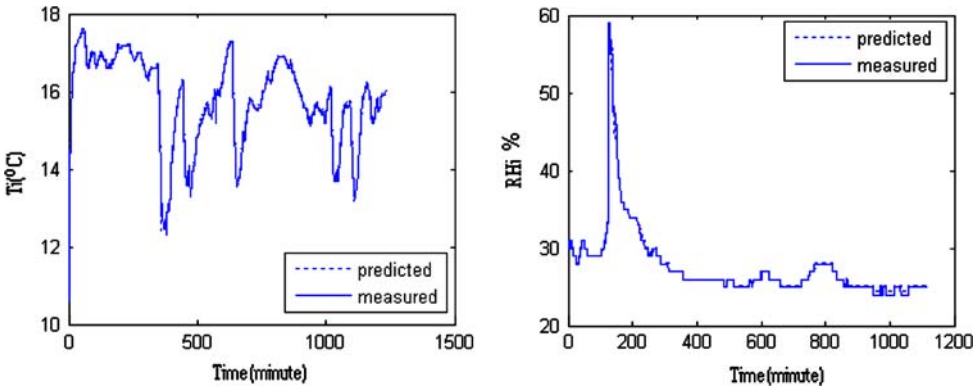

Fig. 5 The comparison between predicted indoor temperature and measured (*left*) and the comparison between predicted indoor relative humidity and measured (*right*) (Genetic algorithm) (all time intervals are 15 min)

Table 4 Performance comparison of indoor temperature prediction between networks

Input signals	One-step ahead prediction	Two-step ahead prediction	Three-step ahead prediction	Four-step ahead prediction
$T_i(t-1)$, $T_i(t-2)$ (NNARX)	Testing: MSE = 0.0062 MAE = 0.0481 Training: MSE = 0.0185 MAE = 0.0478 Average generalization error: 0.406	Testing: MSE = 0.0162 MAE = 0.0827	Testing: MSE = 0.0313 MAE = 0.1138	Testing: MSE = 0.0533 MAE = 0.1461
$T_i(t-1)$, $T_i(t-3)$, $T_i(t-5)$ (Genetic algorithm)	Testing: MSE = 0.0080 MAE = 0.0561 Training: MSE = 0.023 MAE = 0.0608 Average generalization error: 0.0139	Testing: MSE = 0.029 MAE = 0.1103	Testing: MSE = 0.057 MAE = 0.2137	Testing: MSE = 0.0909 MAE = 0.2501

- The best combinations for input variables seem not to reflect their physical insights. For instance, indoor temperature is more or less impacted by outdoor temperature in physical sense. However, in network models, outdoor temperature has no significant impact on indoor temperature. This might be because of well-insulated and tightly-sealed building envelope that outdoor temperature has little impact on indoor temperature. Or the mechanical activities inside the test house overwhelmed natural activities.
- Linear-unlearning-leave-one-out (LULOO) technique [20] does not work well in our work. Tables 4 and 5 illustrate that indoor relative humidity prediction with genetic algorithm performs better than indoor temperature prediction with NNARX. However, other validation criteria give an opposite conclusion. It is unknown what may cause LULOO malfunction. The above result demonstrates LULOO's unreliability. As mentioned in Sect. 3.3.1, LOO [10] method is more accurate and reliable. We also tried this method in this study. However, due to the large size of our training data, almost 2000, the heavy computational burden made us have to give it up. Bootstrap [21] is another accurate resampling method and requires less calculation than LOO. But as we have four neural networks for validation, Bootstrap needs at least hundred times of retraining for each network and the total work for estimation of the average generalization error is still considerably huge. Therefore Bootstrap was not considered this time in our work. In practice, the use of these resampling methods often causes dilemma. On the one hand, due to computational load, these resampling methods seem to be more suitable for small-size samples. On the other hand, the accuracy and reliability of estimation require a large number of retraining when employing these two resampling methods. In addition,

Table 5 Performance comparison of indoor relative humidity prediction between networks

Input signals	One-step ahead prediction	Two-step ahead prediction	Three-step ahead prediction	Four-step ahead prediction
$RH_i(t-1)$, $T_i(t-2)$, $T_o(t-2)$, $RH_o(t-1)$ (NNARX)	Testing: MSE = 0.239 MAE = 0.14 Training: MSE = 0.0805 MAE = 0.0974 Average generalization error: 0.0429	Testing: MSE = 0.7331 MAE = 0.2568	Testing: MSE = 1.3309 MAE = 0.3562	Testing: MSE = 1.9242 MAE = 0.4445
$RH_i(t-1)$, $RH_i(t-2)$, $RH_i(t-3)$, $RH_i(t-5)$ (Genetic algorithm)	Testing: MSE = 0.1584 MAE = 0.1717 Training: MSE = 0.0446 MAE = 0.0624 Average generalization error: 0.0322	Testing: MSE = 0.3617 MAE = 0.4262	Testing: MSE = 0.822 MAE = 0.7255	Testing: MSE = 1.5021 MAE = 0.9996

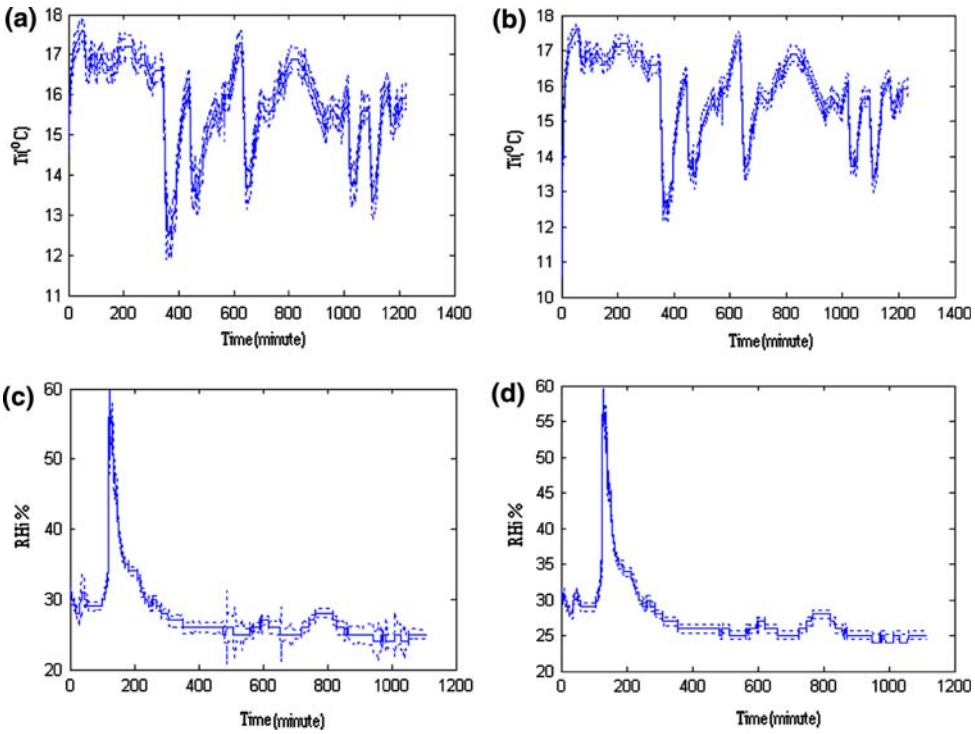

Fig. 6 **a** 95% prediction intervals for indoor temperature prediction (NNARX), **b** 95% prediction intervals for indoor temperature prediction (Genetic algorithm), **c** 95% prediction intervals for indoor relative humidity prediction (NNARX), **d** 95% prediction intervals for indoor relative humidity prediction (Genetic algorithm) (all time intervals are 15 min)

the main goal of estimating the average generalization error is for model selection which is a time-consuming work. Although searching techniques have been improved very much nowadays, more work is still needed. Take some popular searching methods as an example, for instance the genetic algorithm. Very often in a modern personal PC, it may take one or even more days to obtain an optimal structure for a neural network when applying a genetic algorithm. These facts require that the methods on the estimation of the average

Table 6 Results for 95% prediction intervals

	T_i (NNARX)	T_i (Genetic algorithm)	RH_i (NNARX)	RH_i (Genetic algorithm)
The number of testing Points	1,228	1,228	1,118	1,118
Average interval width (full)	0.5945	0.4797	1.8677	1.1844
The number of targets which don't fall in the prediction interval	6	22	55	76

generalization error should be fast, reliable and relatively accurate. Therefore, to find a fast and reliable method on the estimation of the average generalization error or to improve LULOO technique for large-size sample data is one of the most important directions for future work.

- If all information of a dynamic system is extracted by a network, its residual (i.e. prediction errors) should be random which are uncorrelated with past errors as well as all past inputs. Residual analysis intends to investigate this. Despite the importance of residual analysis, in our case it is almost certain that residuals would have some degree of correlation with past data and inputs as important information, heating power and ventilation rate, were missing. Therefore, residual analysis was not taken as validation criterion in this work.
- The prediction intervals are displayed in Fig. 6 and Table 6.

Table 6 shows that both models for indoor temperature prediction possess very good prediction intervals, in which only few targets are not in intervals. On the other hand, for indoor relative humidity prediction, many targets are out of prediction intervals. Probably this is because the system of indoor relative humidity is more dynamic and uncertain as we described previously and some information is missing for the networks. However, the exact binomial distribution shows that the number of targets outside 95% prediction intervals will probably lie (92.6% probability) in the range between 45 and 80. Therefore, prediction-interval Eqs. (13, 14) still work well for indoor relative humidity prediction. Moreover, these results also provide more evidence that indoor relative humidity is more difficult to predict as big error bars were obtained for indoor relative humidity predictions. Veanux et al. [22] has proved that Eq. (14) gives more accurate prediction than Eq. (13). This is true because both genetic models have narrower intervals than NNARX models. It is very obvious that Eq. (13) underestimates prediction interval which is the reason why many targets are outside prediction intervals from genetic models.

- In Fig. 6c, some areas have much wider prediction intervals, which shows uncertainties in these areas for the network from NNARX. On the other hand, the network from genetic algorithm (Fig. 6d) has more reliable predictions from almost all areas.

5 Conclusions

Four three-layer feedforward neural networks were developed by NNARX and genetic algorithm. A rich and detailed validation investigation, including prediction interval and *k*-step ahead prediction, was made as well. Temperature predictions got very accurate results. Correlation coefficients between predicted values and measured are 0.998 and 0.997 for NNARX and genetic models, respectively (see Fig. 7). Indoor relative humidity prediction could achieve satisfactory precision with correlation coefficient 0.997 (see Fig. 7). Prediction intervals were estimated by fast adopted delta method. Despite the possible inaccuracy of delta method, it is still a very practical way to examine the uncertainty inherent in the predictions in terms of weights and biases. Preliminary results showed that three-layer feedforward neural network is capable of approximating any nonlinear relations even in a complex situation where some impact factors are still unclear and some important information is unavailable, and NNARX model is particularly workable on engineering purpose. Genetic algorithm is a sufficient search algorithm for structure selection as long as proper initial population and reproduction rules are set. But still there is more work need to be done.

First of all, the accuracy for indoor relative humidity prediction needs to be improved. One possibility to achieve this is to use neural network ensemble techniques [12] [25] to set up a separate network to estimate noise variance and then combine genetic algorithm to select optimal structure. The ensemble technique can also be used to capture extreme values, which are possibly influenced by high noises or other unknown factors. By now, when the measured indoor relative humidity is above 50%, predicted values do not fit observed values well.

Secondly, linear-unlearning-leave-one-out (LULOO) [20] technique for average generalization error prediction does not work as well as it could in our work. Further investigation and improvement on LULOO will be our

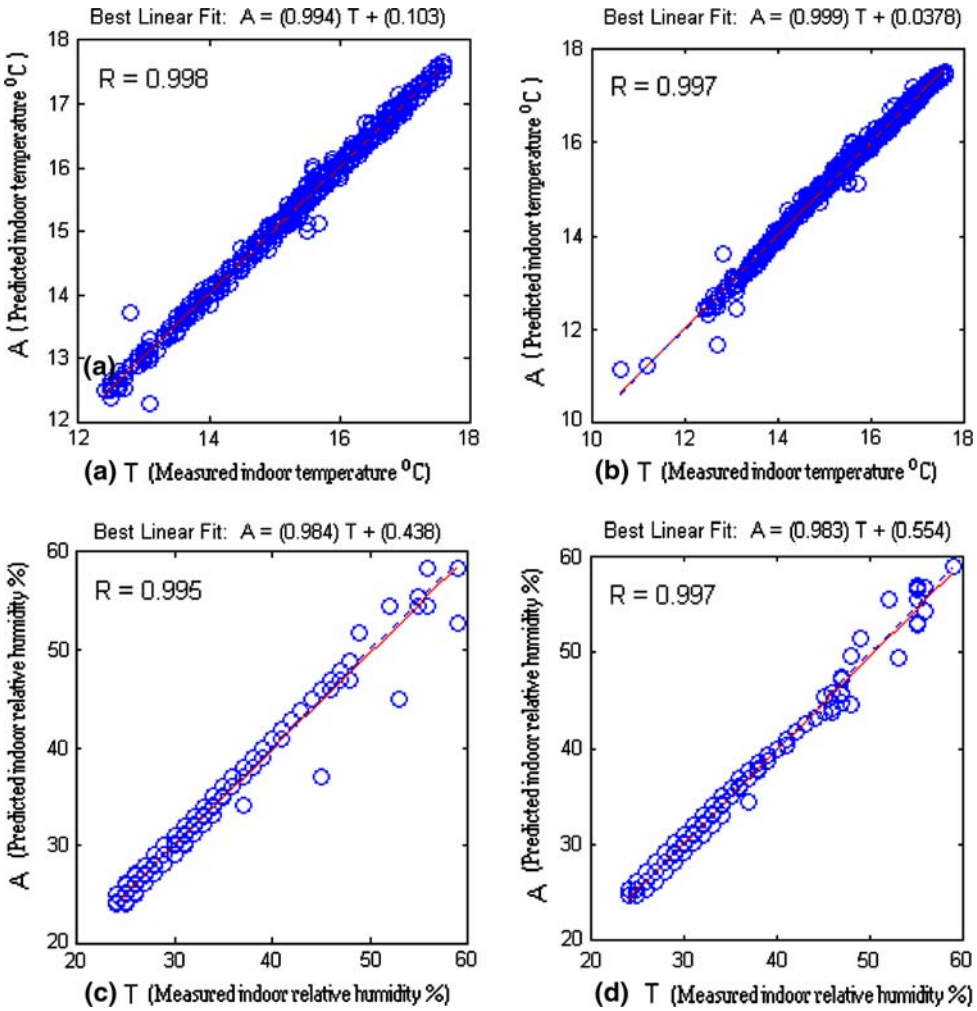

Fig. 7 Post-regression results between predicted data and measured. **a** Indoor temperature prediction (NNARX), **b** Indoor temperature prediction (Genetic algorithm), **c** Indoor relative humidity prediction (NNARX), **d** Indoor relative humidity prediction (Genetic algorithm)

future work. The ultimate goal is to combine fast LULOO technique with genetic algorithm for model selection.

Finally, we will extend our work to more complicated case for building control, detection and prevention practices.

Acknowledgments We are grateful to the Academy of Finland for financial support.

References

1. Reijula K (2004) Moisture-problem buildings with molds causing work-related diseases. Adv Appl Microbiol 55:175–189
2. Luosujarvi R, Husman T, Seuri M, Pietikainen M, Pollari P, Pelkonen J, Hujakka H, Kaipiainen-Seppanen O, Aho K (2003) Joint symptoms and diseases associated with moisture damage in a health center. Clin Rheumatol 22:381–385
3. Lu X (2002) Modelling heat and moisture transfer in buildings—(I) model program. Energy Build 34:1033–1043
4. Teodoisu C, Hohota R, Rusaouën G, Woloszyn M (2003) Numerical prediction of indoor air humidity and its effect on indoor environment. Build Environ 38(5):655–664
5. Ruano AE, Crispim EM, Conceição EZE, Lúcio MMJR (2006) Prediction of building's temperature using neural networks models. Energy Build 38:682–694
6. Sigumonrong AP, Bong TY, Fok SC, Wong YW (2001) Self-learning neurocontroller for maintaining indoor relative humidity. In: Proceedings of the International Joint Conference on Neural Networks v2, IEEE, Washington, DC, USA, pp 1297–1301
7. Zhang Q, Wong YW, Fok SC, Bong TY (2005) Neural-based air-handling unit for indoor relative humidity and temperature control. In: ASHRAE Transactions v 111 PART 1—Technical and Symposium Papers presented at the 2005 Winter Meeting of the American Society of Heating, Refrigerating and Air-Conditioning Engineers, ASHRAE, Orlando, FL, USA, pp 63–70
8. Ferreira PM, Faria EA, Ruano AE (2002) Neural network models in greenhouse air temperature prediction. Neurocomputing 43:51–75
9. Thomas B, Soleimani-Mosheni M (2007) Artificial neural network models for indoor temperature prediction: investigations in two buildings. Neural Comput Appl 16:81–89
10. Nørgaard M, Rvan O, Poulsen NK, Hansen LK (2000) Neural networks for modelling and control of dynamic systems. Springer, London
11. Tibshirani R (1996) A comparison of some error estimates for neural network models. Neural Comput 8:152–163

12. Heskes T (1997) Practical confidence and prediction intervals. In: Mozer M, Jordan M, Pekes T (eds) Advances in neural information processing system 9. MIT Press, Cambridge, pp 176–182
13. Chen S, Billings SA, Cowan CFN, Grant PM (1990) Practical identification of Narmax models using radial basis functions. Int J Control 52:1327–1350
14. Irie B, Miyaki S (1988) Capabilities of three layer perceptrons. In: Proceedings of the IEEE Second International Conference on Neural Networks, San Diego, CA
15. Demuth H, Beal M (1988) Neural network toolbox user's guide. Version 3.0. The Math Works, Inc. Natick
16. Bloem H (1993) Workshop on system identification applied to building performance data. Institute for systems engineering and informatics, Joint research centre, Ispra, Italy
17. Levenberg K (1944) A method for the solution of certain problems in least squares. Q Appl Math 2:164–168
18. Marquard D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11:431–441
19. Nørgaard M (2000) Neural network based system identification toolbox. Tech. Report. 00-e 891, Department of Automation Technical University of Denmark
20. Hansen LK, Larsen J (1996) Linear unlearning for cross-validation. Adv Comput Math 5:269–280
21. Efron B, Tibshirani R (1993) An introduction to the bootstrap. Chapman & Hall, New York
22. De Veanux RD, Schumi J, Schweinsberg J, Ungar LH (1998) Prediction intervals for neural networks via nonlinear regression. Technometrics 40(4):273–282
23. Bishop CM (1995) Neural Networks for pattern Recognition. Clarendon Press, Oxford
24. Schaffer JD, Whitley D, Eshelman L (1992) Combination of genetic algorithm and neural networks: A survey of the state of art. In: International workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, MD, USA, pp 1–37
25. Sharkey AJC (1999) Combining artificial neural nets: ensemble and modular multi-net systems. Springer, Berlin