Relearning procedure to adapt pollutant prediction neural model: Choice of relearning algorithm

Philippe Thomas, Marie-Christine Suhner, William Derigent

2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, July 14-19, 2019

Relearning procedure to adapt pollutant prediction neural model: Choice of relearning algorithm

Philippe Thomas
CRAN UMR 7039
Université de Lorraine
Vandœuvre, France
philippe.thomas@univ-lorraine.fr

Marie-Christine Suhner
CRAN UMR 7039
Université de Lorraine
Vandœuvre, France
marie-christine.suhner@univ-lorraine.fr

William Derigent
CRAN UMR 7039
Université de Lorraine
Vandœuvre, France
william.derigent@univ-lorraine.fr

Abstract. Predict the indoor air quality becomes a global public health issue. That's why Airbox lab® company develops a smart connected object able to measure different physical parameters including concentration of pollutants (volatile organic compounds, carbon dioxide and fine particles). This smart object must embed prediction capacities in order to avoid the exceedance of an air quality threshold. This task is performed by neural network models. However, when some events occur (change of people's behaviors, change of place of the smart connected object as example), the embedded neural models become less accurate. So a relearning step is needed in order to refit the models. This relearning must be performed by the smart connected object, and therefore, it must use the less computing time as possible. To do that, this paper propose to combine a control chart in order to limit the frequency of relearning, and to compare three learning algorithms (backpropagation, Levenberg-Marquardt, neural network with random weights) in order to choose the more adapted to this situation.

Keywords—indoor air quality, neural network, relearning, control chart, NNRWs, backpropagation, Levenberg-Marquardt.

I. INTRODUCTION

In industrialized countries, the air quality is become a real preoccupation of population and governments in the last decades. However, in people's mind, this problem is mainly related to the outdoor quality. As example, an opinion survey indicates that nine out of ten people consider as good the indoor quality of their housing even if they are not able to evaluate it [1]. Moreover, the US Environmental Protection Agency estimates that the indoor pollutants levels are 2 to 5 times higher than outdoor ones [2]. The main pollutants are biological (bacterium, viruses), chemical (volatile organic compounds (VOCs), carbon monoxide (CO)), fine particles, radioactive gas, tobacco, humidity. Spengler and Sexton [3] have listed the human health impacts of these different pollutants. The pollution sources are various, including building material, furniture, cleaning products, outdoor air.

Considering these facts, people need tools to evaluate the indoor air quality (IAQ) [1]. That's why Airbox lab® company develops a smart connected object called Foobot® [4]. Foobot® can measure five physical quantities every minute: temperature, humidity, concentration of VOC, concentration of carbon dioxide (CO_2), and fine particles (pm). These data are stored in the Foobot® itself or may be collected in a distant database. Beyond this goal of measure and data storing, Airbox lab® want to embed prediction capacities in their Foobot® in order to be able to forestall an increasing of pollutant levels. Neural models have been proposed in the past to detect the current situation (cooking, sleeping as example) [5]. These current situations are used as inputs in another neural model which can predict the evolution of pollutants levels (30 minutes forecast) [6].

However, the performances of these models may be greatly degraded when different events occur (change of people's behaviors, change of place of the Foobot® as example). In this case, a relearning step must be performed. However, the relearning must be performed by the Foobot® which is a smart object including small computing power. So, to reduce the computing power need, relearning step must be performed only when it is necessary. To do that, a control chart may be used [7]. Moreover, the relearning itself must use least possible computing time.

The learning of neural network is mainly performed by using gradient based methods including first (backpropagation [8]) or second order approach (Levenberg-Marquardt [9, 10]). However, since the pioneering works of Schmidt et al. [11] an alternative approach is to use neural network with random weights (NNRWs) where weights and biases connecting the input to the hidden layers are randomly chosen and only the parameters connecting the hidden to the output layers are updated by the algorithm. This approach has been successively applied in many applications [12, 13]. Within this philosophy, we propose to adapt only the parameters connecting the hidden to the output layers when relearning is needed.

The main goal of this paper is to evaluate these three algorithms in terms of accuracy, but also and above all in terms of calculation cost in order to adapt Foobot® embedded neural network to new situation.

In the following section, a short state of the art about the prediction of indoor air pollution using computational intelligence is presented. Part 3 recalls the notations and algorithms used to learn the initial model. In part 4, the control chart used to determine when a re-learning is needed, is recalled. Section 5 recalls the NNRWs structure and learning algorithm used for the re-learning step. Part 6 is devoted to the study of the computing cost of the three learning algorithms to compare. Section 7 presents the industrial application, and the results obtained during the adaptation process before to conclude.

II. STATE OF THE ART

The IAQ is an important issue concerning many researches in different field during the last years. Organizations, such as World Health Organization, list the different pollutants able to impact the IAQ and define thresholds not to be exceeded [14-15]. Pollutants impacting IAQ are generally grouped in three main categories [16]:

- Outdoor air pollutants (CO, benzene, ozone, oxides of nitrogen)
- Pollutants generated by occupants' activities ($C0_2$, particle matter (pm))
- Pollutants generated by building material and furniture (volatile organics, microbial contaminants)

The monitoring of outdoor air quality is performed by many agencies as ATMO [17]. Challoner et al. [18] propose to exploit this sort of information in order to evaluate the level of dioxide of nitrogen and pm in commercial building without using additive sensor. They propose to model the link between outdoor air quality and IAQ by using artificial neural network. Their results show good performance for the dioxide of nitrogen level prediction, but the prediction model for pm is not efficient.

Another approach to monitor these pollutants is to develop and use sensors adapted to these different pollutants [19]. A recent approach is to propose to use a "do it yourself" approach based on the use of Arduino® microcontroller [20]. The data collected by these sensors have been used with different objective. Tijani et al. [21] use the concentration of CO_2 to simulate occupants' behaviors in office building. They use a dynamic Bayesian network able to use conjointly expert knowledge and dataset. Other authors try to use the IAQ to predict the sick building syndrome [22]. They develop an artificial neural network model (ANN) able to link different surveyed pollutants, including CO_2, pm, VOC, airborne bacteria and fungi, to an occupant symptom metric. Another approach is to use the pollutants levels collected by sensors to classify the sources influencing IAQ like fragrance presence, foods and beverages, human activities as window opening, by using ANN classifiers [23-24].

Predict the evolution of air quality is another objective that can be pursued. Abd Rahman et al. [25] proposed a forecasting of outdoor air quality. They compare three time series models (Autoregressive integrated moving average, ARIMA, ANN and fuzzy time series FST) and concluded that ANN is the most suitable model. With the same philosophy, forecasting model has been proposed to predict the IAQ. Yu and Lin [26] use an ARIMA model and exploit temperature, humidity and CO_2 concentration data when Thomas et al. [6] use an ANN model and exploit temperature, humidity and CO_2 concentration as well, but also VOC and pm.

III. MULTILAYER PERCEPTRON

A. Structure

Multilayer neural network (MLP) is an ANN incorporating a single hidden layer (using a sigmoidal activation function). Its structure (for single output case) is given by:

$$o = \sum_{h=1}^{m_h} w_h . g\left(\sum_{i=1}^{m_i} v_{hi} . x_i \right) = \sum_{h=1}^{m_h} w_h . g\left(V_h . X \right) = \sum_{h=1}^{m_h} w_h . H_h \quad (1)$$

where x_i are the m_i inputs (x_{m_i} is a constant input equal to 1 in order to that v_{hm_i} be the bias of the hidden neuron h), v_{hi} are connecting weights between input and hidden layers, $g(.)$ is the activation function of the hidden neurons (hyperbolic tangent), w_h are connecting weights between hidden and output layers (H_{m_h} is a constant equal to 1 in order to that w_{m_h} be the bias of the output neuron), and o is the network output.

B. Learning algorithm

The adaptation of the weights of the MLP is performed by using a local search of the minimum of the classical quadratic criterion:

$$V(\theta) = \frac{1}{2N} \sum_{k=1}^{N} \varepsilon^2(k, \theta) \quad (2)$$

where θ groups together all the network weights, N is the size of the learning dataset and ε is the prediction error:

$$\varepsilon(k, \theta) = t(k) - o(k, \theta) \quad (3)$$

where $t(k)$ is the target value of the pattern k and $o(k, \theta)$ is the predicted one by the network.

The 2nd order Taylor series expansion of the criterion to minimize (2) leads to the classical Gauss-Newton algorithm:

$$\hat{\theta}^{i+1} = \hat{\theta}^i - (He(\hat{\theta}^i))^{-1} V'(\hat{\theta}^i) \quad (4)$$

where $\hat{\theta}^i$ is the estimation of θ at iteration i, $V'(\hat{\theta}^i)$ is the gradient of the criterion:

$$V'(\theta) = -\frac{1}{N} \sum_{k=1}^{N} \psi(k, \theta) . \varepsilon(k, \theta) \quad (5)$$

where $\psi(k, \theta)$ is the gradient of $o(k, \theta)$ with respect to θ.

$He(\hat{\theta}^i)$ is the Hessian matrix. The Levenberg-Marquardt update rule allows to estimate it:

$$He(\theta) = \frac{1}{N}\sum_{k=1}^{N}\psi(k,\theta).\psi^{T}(k,\theta) + \beta I \quad (6)$$

where I is the identity matrix and β a small non negative scalar which must be adapted during the learning process. A robust version of this algorithm has been proposed by Thomas et al. [27].

When the Taylor series expansion of the criterion to minimize (2) is performed to the 1st order, this leads to the classical backpropagation algorithm:

$$\hat{\theta}^{i+1} = \hat{\theta}^{i} - \eta.V'(\hat{\theta}^{i}) \quad (7)$$

where η is the learning rate.

IV. Relearning need assessment process

When batch learning approach is used, the obtained model is frozen when the learning is completed. However, in many cases, the system which is modelled continues to live and evolve. This is the case, when the system is worn (tool of a milling machine as example) or if the system is modified (part replacement for reparation or enhancement as example) or if the environment is modified (system relocation as example). In this case, a drift occurs between the frozen model and the living system. With the time, the model becomes less accurate and the model must be refitting by using relearning approach.

However, relearning is time consuming and must be performed only if needed. Noyel et al. [7] have proposed to use a control chart [28] in order to determine when relearning is needed.

A. Control chart

Control chart is one of the seven basic tools for quality control. Its main goal is to determine if a production process remains under control. From a statistical point of view, when the production process is under control, the risk of a point exceeding a 3σ control limit (Pareto) is of 0.27%. This control chart may be used to determine if a drift occurs between the model and the system behaviors. During the design of the initial neural model, the database has been divided into learning and validation dataset. The learned model has been tested on the validation dataset and the error performed is characterized by its distribution (supposed gaussian) its mean (expected zero to avoid systematic error) and its standard deviation σ.

When the model is used to monitor the system, new data are collected, and an error is performed, principally characterized by its standard deviation σ_n. The main goal of control chart is to determine if σ_n remains near σ or not. To do those, two bounds must be determined. These bound are called the Upper Center Line (*UCL*) to determine if σ_n is too great to be considered as statistically equivalent to σ and Lower Center Line (*LCL*) to determine if σ_n is too small to be considered as statistically equivalent to σ. In our case, the fact that σ_n is lower than *LCL* is not a problem and so, only *UCL* is considered. The determination of σ_n is performed on a time window of size n. So, each point in the control chart represents the variance of the error obtained on a dataset of size n.

B. Control bounds

As explained part 2.1., Only the *UCL* is of interest in the considered case. This limit is calculated to represent 99.8% of data [26]:

$$UCL = CL + 3\frac{CL}{c_4}\sqrt{(1-c_4^2)} \quad (8)$$

with c_4 given by:

$$c_4 = \sqrt{\frac{2}{n-1}}\frac{\left(\frac{n}{2}-1\right)!}{\left(\frac{n-1}{2}-1\right)!} \quad (9)$$

where n is the size of the sample corresponding to the considered time window.

CL is an unbiased estimated standard deviation of the error performed on the validation dataset during the learning step:

$$CL = c_4.\sigma \quad (10)$$

The monitored characteristic which must be compared to UCL is standard deviation of the error performed by the network on the sample corresponding to the time window of size n:

$$s = \sqrt{\frac{\sum_{k=1}^{n}\left(\varepsilon_k - \bar{\varepsilon}\right)}{n-1}} \quad (11)$$

where ε_k is the error performed on data k and $\bar{\varepsilon}$ is the mean of the error performed on the sample of size n.

If the value of s is lower to UCL for a sample, no relearning is needed. If it is upper to this limit, the neural model must be adapted by performing a relearning.

V. Relearning algorithms

The initial model is a single output multilayer perceptron (MLP) learned on an initial dataset as explain part III and its structure is given by (1). The matrix writing of equation (1) is:

$$O=W.H \quad (12)$$

All the weights of this MLP has been learned by using a classical learning algorithm (here a robust version of the Levenberg-Marquardt algorithm [24]).

When the control chart described at part IV indicates that a relearning is needed, these weights must be updated. This task may be performed by using again the same Levenberg-Marquardt algorithm [9, 10], or it can be replaced by a simplest gradient algorithm [8]. Another approach may be to retrain only the weights connecting the hidden to the output layers, within the same philosophy than the algorithm NNRWs proposed by Schmidt et al. [11], the main idea here is to fix the weights connecting the input to the hidden layer and to adapt only the weights matrix *W* connecting the hidden layer to the output neuron by:

$$W = H^{\dagger}.T \tag{13}$$

where *T* is the vector of the target (desired output) for the relearning dataset and $H^{\dagger}$ is the Moore-Penrose generalized inverse matrix of *H* [30, 31, 32]. *H* is calculated by using the fixed weights connecting the input to the hidden layers and the inputs of the relearning dataset.

VI. Evaluation of the computing cost

The relearning algorithm must be embedded in a smart object. The computing cost is so an important issue for the choice of the relearning algorithm to use. To select it, the number of basic operations must be evaluated.

A. Basic operations for MLP output evaluation

The output of the MLP is given by (1) where g(.) stands for the hyperbolic tangent which must be evaluated:

$$g(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{14}$$

Knowing that $e^{-z} = 1/e^{z}$, the calculation of this hyperbolic tangent needs 1 subtraction, 1 addition, 3 divisions and the evaluation of one exponential:

$$e^{z} = \sum_{k=0}^{\infty} \frac{z^{k}}{k!} \tag{15}$$

Which must be evaluated by using an iterative approach where each loop uses 4 basic operations (2 additions, 1 multiplication, 1 division). The number of loops depends of the required precision. As example, to obtain a relative error lesser than 10^{-15} to calculate e^{1}, 18 iterations are needed. However, to calculate e^{100}, we need 189 iterations to obtain the same precision. For the sequel, we choose to use a mean of 100 iterations to evaluate the different exponentials. The number of basic operators to evaluate an exponential is so estimated to 400, leading to the fact that the evaluation of an hyperbolic tangent uses on average 405 basic operators.

The evaluation of the hidden neuron output H_h for one hidden neuron and one pattern needs to use m_i multiplications, $m_i - 1$ additions and one hyperbolic tangent leading to the use of $2.m_i + 404$ basic operators. So, the evaluation of the hidden neuron outputs for all hidden neurons and all patterns of the learning dataset needs $(m_h - 1).N.(2.m_i + 404)$ basic operators.

The evaluation of the outputs of the MLP *o* given by (1) for the complete dataset needs $N.(m_h - 1)$ additions, $N.m_h$ multiplications and the evaluation of the $(m_h - 1)$ hidden outputs H_h. So the number of basic operations needed to evaluate the MLP output is:

$$Op_{MLP} = N.m_h\,(2.m_i + 406) + N \tag{16}$$

B. Basic operations for the backpropagation algorithm

The backpropagation algorithm is given by (7). It is an iterative algorithm which evaluates the gradient of the criterion to minimize given by (5).

To evaluate the gradient, we need to determine the gradient of $o(k,\theta)$ with respect to θ: $\psi(k,\theta)$. $\psi(k,\theta)$ is a vector of length corresponding to the number of MLP parameters:

$$m_\theta = (m_h - 1).m_i + m_h \tag{17}$$

The elements of $\psi(k,\theta)$ corresponding to the weights connecting the hidden to the output layers are obtained directly. Those correponding to the $m_h.m_i$ weights connecting the input to the hidden layers need, for each pattern, to perform 2 multiplications and 1 subtraction. So, the evaluation of $\psi(k,\theta)$ needs $3.((m_h - 1).m_i)$ basic operations.

The evaluation of the gradient $V'(\hat{\theta}^i)$ for one pattern *k* needs the evaluation of $\psi(k,\theta)$ and $m\theta$ multiplications. This work must be performed N times and the results must be summed up leading to the use of $m_\theta.(N-1)$ additions and 1 division. So the number of basic operations needed to evaluate the gradient $V'(\hat{\theta}^i)$ for one iteration is:

$$Op_{V'} = \left(\left(3.(m_h - 1).m_i\right) + m_\theta\right).N + (N-1).m_\theta + 1 \tag{18}$$

To evaluate the backpropagation algorithm for one iteration, we need to evaluate the output of the MLP *o*, the gradient $V'(\hat{\theta}^i)$ and to perform m_θ additions and m_θ multiplications. So the number of basic operations needed to perform the backpropagation algorithm is:

$$Op_{BP} = It.(Op_{MLP} + Op_{V'} + 2.m_\theta) \tag{19}$$

where *It* is the number of iterations performed. From this sum, we have excluded the number of operations needed to update the parameter η.

C. Basic operations for the LM algorithm

The LM algorithm is given by (4). It is an iterative algorithm which evaluates the gradient of the criterion and the hessian matrix which must be inverted. The computational cost for the gradient is given by (18).

The Levenberg-Marquardt approximation of the Hessian is given by (6). It needs $N.(m_\theta.m_\theta)$ multiplications and $(N).(m_\theta.m_\theta)$ additions and one division:

$$Op_H = 2.N.m_\theta^2 + 1 \tag{20}$$

This matrix Hessian is a square matrix of dimension (m_θ, m_θ) and must be inverted.

Different algorithms may be used for the inversion of an invertible matrix of dimension (d, d). The one considered here is the use of the Gaussian elimination which need the building of a matrix of dimension (d, $2d$) by adjunction of an identity matrix. In the worst of the cases, this inversion procedure uses $(d-1)$ lines permutations which corresponds to $2.d.(d-1)$ elements permutations which can be considered as basic operations too. Globally, the matrix inversion procedure uses:

$$Op_{inv} = 4.d.\left(d^2 - 1\right) \tag{21}$$

For the Hessian matrix, this corresponds to:

$$Op_{invH} = 4.m_\theta.\left(m_\theta^2 - 1\right) \tag{22}$$

So the number of basic operations needed to perform the backpropagation algorithm is:

$$Op_{LM} = It.\left(Op_{MLP} + Op_{V'} + Op_H + Op_{invH} + m_\theta^2 + m_\theta\right) \tag{23}$$

where *It* is the number of iterations performed. From this sum, we have excluded the number of operations needed to update the parameter β.

D. Basic operations for the NNRWs algorithm

The NNRWs algorithm is given by (13). It implies to use the Moore-Penrose inverse given by:

$$H^\dagger = H^T.\left(H.H^T\right)^{-1} \tag{24}$$

The product of the matrices $H.H^T$ needs $(2.m_h - 1).N^2$ basic operations and produce a matrix of dimension (N, N). This matrix must be inverted and (21) implies that $4.N.(N^2-1)$ basic operations must be used. The Moore-Penrose inverse needs Op_{MP} basic operations:

$$Op_{MP} = (2.m_h - 1).N^2 + 4.N.(N^2 - 1) + (2.N - 1).N.m_h \tag{25}$$

So, the number of basic operations needed to perform the NNRWs algorithm is:

$$Op_{NNRWs} = Op_{MLP} + Op_{MP} + (2.N - 1).m_\theta \tag{26}$$

In comparison with the two other algorithms, this algorithm is not iterated, and don't need the tuning of a parameter.

VII. Industrial Application

A. Description of the case study

An inhabited private house has been instrumented by 5 Foobot® located in different rooms of the house. Figure 1 presents the implementation plan of the 5 Foobot®.

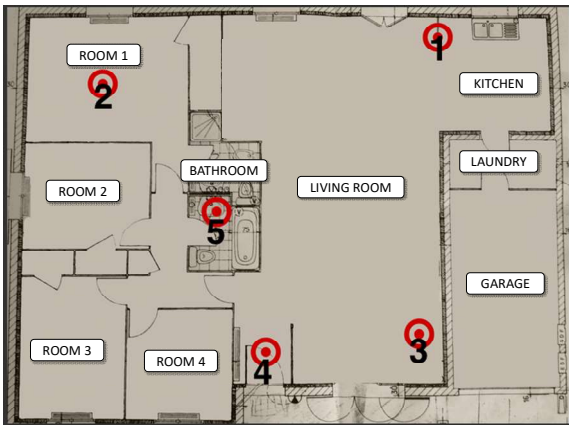

Fig. 1. Implantation plan of the Foobot®.

For one month the inhabitants of the house have indicated their occupations (cooking, sleeping, house cleaning as example). In a first step, classification models have been designed in order to predict these occupations. The outputs of these models are used as inputs of a regression neural network model which predicts the evolution of pollutants levels (30 minutes forecast) (figure 2). These models use also as inputs the past values of the five collected physical quantities: temperature (T°), humidity (Hu), concentration of VOC (VOC), concentration of CO_2, formaldehyde and fine particles (pm). A first work [6] have allowed to design these prediction neural network models by using Foobot® 1 (Kitchen) dataset. The optimal structure of the models is obtained by using pruning step [33].

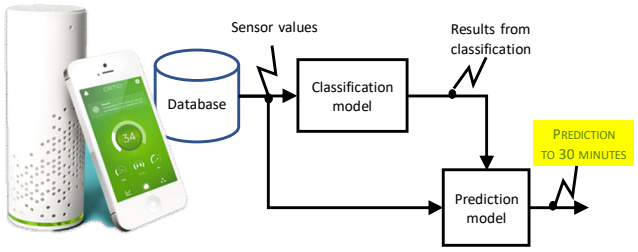

Fig. 2. Principe of pollutants level prediction.

The resulting models are accurate and allow to predict the evolution of pollutants level with 30 minutes forecasting.

B. Refitting of prediction neural models

The preceding neural model has been built with the dataset collected by the Foobot® 1 located in the kitchen of the considered house. This model is suitable to predict the evolution of pollutant levels by using surveyed data collected by Foobot® 1. However, when this model is used with the dataset collected by Foobot® 2 located in the bedroom, the results are no longer relevant. So, Foobot® must embed neural model refitting capability to adapt the model to the local condition. This refitting capability includes the capabilities to detect that a relearning step is needed, and the relearning itself.

Figure 3 presents the control chart that allows to determine when relearning is needed. The LCL is not represented because it is not used here. The control chart obtained with the Levenberg-Marquardt learning algorithm (LM) is given by red circles. The control chart obtained with the back propagation (BP) algorithm is given by black stars. The control chart obtained with the NNRWs algorithm is given by blue crosses.

The center line (CL) which corresponds to the standard deviation of the error obtained during the validation is indicated in green. The upper center line (UCL) given by equation (1) is indicated in red. Blue crosses (respectively red circles and black stars) represent the standard deviation of the error performed on the considered sample when NNRWs (respectively LM and BP) relearning algorithm is used. The size n of the samples is fixed to 120 data that corresponds to a time window of 2 hours.

The initial model is those built with Foobot® 1 dataset to predict the CO_2 concentration level. When this model is used with the first sample of dataset given by Foobot® 2, the standard deviation of the error is outside the acceptable limits (upper to the UCL) and so, a relearning is needed. This fact shows that the model built with Foobot® 1 dataset is not suitable for the dataset collected by Foobot® 2. However, the relearning step for the three algorithms allows to refit the model, and samples 2 and 3 are under the UCL. It can be noticed that this relearning after sample 1 is not enough because standard deviation of the error obtained on samples 4, 5, 8, 9, ,12, 13, 14, 15, 16 are outside the acceptable limits (upper to UCL). So, after each of these samples, a relearning is performed. After the sample 16, only one sample (36) is outside the acceptable limit. So, we can consider that the relearning performed after samples 4 to 16 are enough to refit the model. These samples correspond to a period of 32 hours, less than 2 days.

Moreover, it can be noticed that the number of relearning needed to refit the model is the same (11) for the three learning algorithms. Moreover, the standard deviation of the error obtained for the three learning algorithms are close together for most of the samples. Only for samples 9, 14 and 15 the values obtained with the NNRWs learning algorithm are significantly different from the values obtained with the two others. This fact shows that the use of NNRWs algorithm in order to refit only the weights connecting the hidden layer to the output neuron is relevant.

All the algorithms were implemented in MATLAB® 2017 and executed on a PC with intel i7 2.90 GHz processor, 32GB of RAM, running the Windows® 10 pro operating system.

Table 1 presents the computing time used to refit the model with the three algorithms. This work has been performed 10 times and the mean and standard deviation of the computing time are presented. This table shows that the use of NNRWs algorithm allows to save many computing times. On average, the computing time used to refit the model with NNRWs algorithms is 14 times shorter than using BP algorithm and 28 times shorter than using LM algorithm. This fact is crucial when this step must be performed by smart object that embed small computing power.

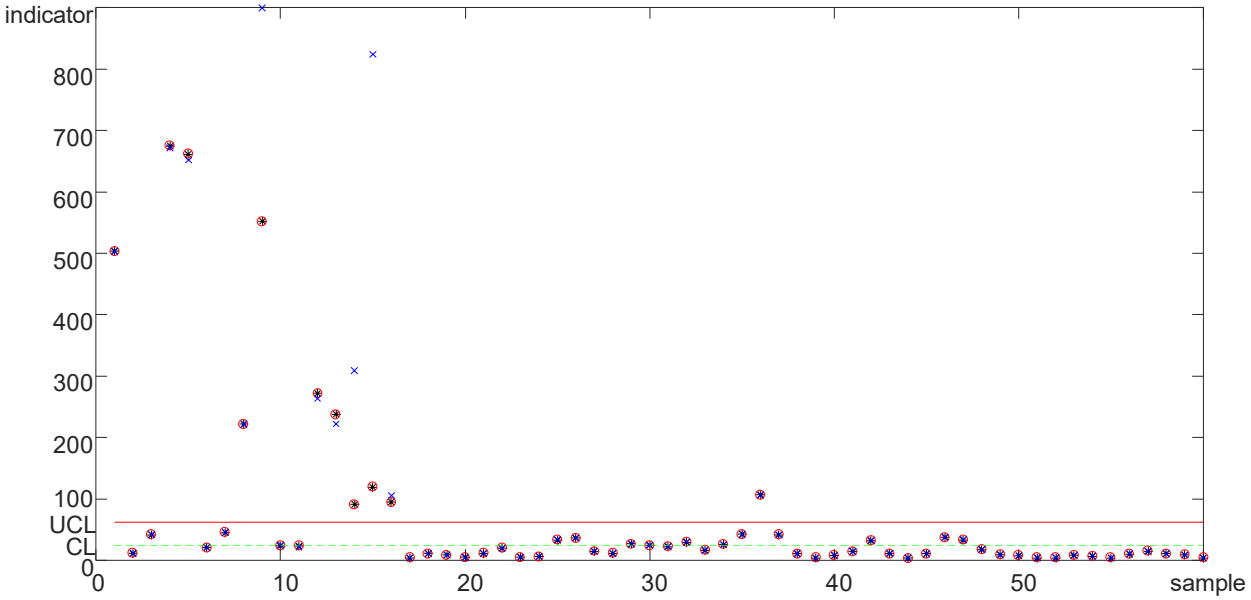

Fig. 3. Control chart for neural model monitoring – blue cross: NNRWs algorithm – red circle: LM algorithm – black star: BP algorithm.

TABLE I. COMPUTING TIME FOR THE RELEARNING STEP.

		LM algorithm	BP algorithm	NNRWs algorithm
Computing time (s)	Mean	0.2712	0.1399	0.0097
	StD	0.0218	0.0079	0.0007

However, this comparison of computing time is not enough because CPUs like i7 implements a state-of-the-art mathematical instruction set that can boost the performance of algorithms that suit this instruction set.

In part VI, the computing cost of the different algorithms used have been studied. These costs depend on the number of parameters (inputs number, hidden neurons number), of the size of the learning dataset, and of the number of iterations performed for the BP and LM algorithms. In the considered case, there are 10 inputs (plus the one for the biases) 3 hidden neurons (plus the one for the biases). The relearning dataset is fixed to 240 and the maximal iteration number is fixed to 200 for BP and to 50 for LM. It can be noticed that supplementary stop criteria are used in order to stop the learning if the improvement of the parameters is too small between two iterations.

These values lead to a computing cost of 33 919 600 basic operations for BP algorithm, 35 097 750 basic operations for LM algorithm and 7 231 883 basic operations for NNRWs algorithm. These values show that NNRWs algorithm uses approximately 4.5 times fewer basic operations than the two others for one relearning step. There exists an important difference with these values and the computing time in table 1 where NNRWs is 14 (respectively 28) times speeder than BP (respectively LM). This fact can be explained by:

- The use of an i7 processor as explained previously,
- The approximation performed for the exponential (15),
- The approximation performed for the matrix inversion (21),
- The other stop criteria used for BP and LM,
- The adaptation of parameters β (resp. η) in LM (resp. BP) not considered.

So, in the considered application, the use of NNRWs algorithm is justified.

However, this choice depends of different parameters. Figure 4 shows the evolution of the computing cost for these three algorithms when the number of iterations performed by BP and LM algorithm evolves between 1 and 200. This figure shows that each iteration of BP algorithm needs less basic operations than LM, but it needs more iterations than LM to converge.

Figure 5 shows the evolution of the same computing cost for the three algorithms when the size of the relearning dataset evolves from 100 to 300 patterns.

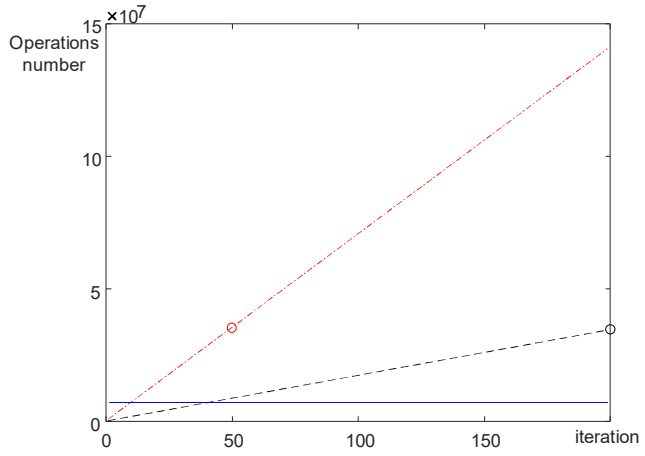

Fig. 4. Evolution of the computing cost in function of iterations number – blue continue: NNRWs algorithm – red dotted: LM algorithm – black dashed: BP algorithm.

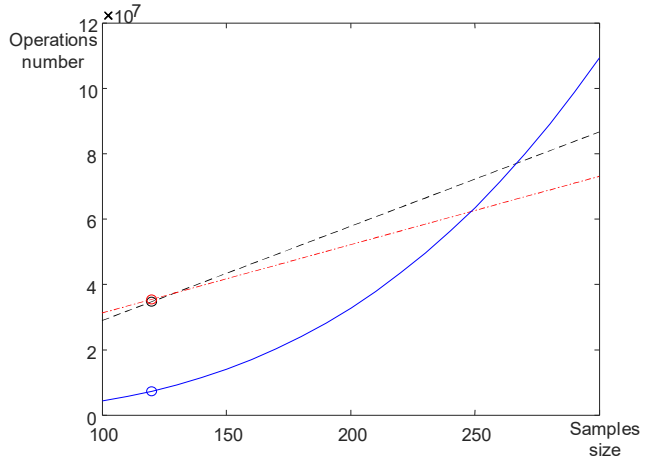

Fig. 5. Evolution of the computing cost in function of the samples size – blue continue: NNRWs algorithm – red dotted: LM algorithm – black dashed: BP algorithm.

This figure shows that the evolution of the computing cost for NNRWs algorithm is exponential when it is linear for the two other algorithms. This fact implies that the use of NNRWs algorithm is pertinent when the relearning dataset is small, but its computing cost explodes when the size of the dataset grows. This is due to the Moore-Penrose inverse algorithm which needs to produce and invert a matrix of dimension (N, N).

VIII. Conclusion

This paper considers the problem of adaptation of neural network models to change. To adapt model to new conditions, a relearning step must be performed. However, relearning step is time consuming. In case of model embedded in smart connected products which include small computing problem, this relearning may be problematic, and its computing time must be reduced. To do that, a combination of a control chart and a particular learning algorithm is investigated. The control chart allows to trigger the relearning only when needed. The learning algorithm used focuses on the adaptation of weights connecting the hidden layer to the output neurons, the other parameters remaining unchanged. Such approach allows to maintain the accuracy of the model while reducing computing time. Now, the relearning step is performed by using all data available since the last relearning. To reduce the computing time, future work may focus on the determination of the size of the optimal relearning dataset.

Acknowledgments

The authors thank the society Airbox lab® for their active financial and scientific support for this work.

References

[1] Harris interactive, http://harris-interactive.fr/opinion_polls/la-perception-de-la-qualite-de-lair-interieur-en-france-aux-etats-unis-en-allemagne-et-au-royaume-uni/, last accessed 2018/05/22 (2013).

[2] Notre planete, https://www.notre-planete.info/environnement/pollution_air/pollution-air-interieur.php, last accessed 2018/05/22 (2018).

[3] Spengler, J. D., Sexton, K.: Indoor air pollution: a public health perspective. Science, 221, 9-17 (1983).

[4] Airbox lab, https://foobot.io/features/, (2003).

[5] Thomas P., Derigent W., Suhner M.C.: Un ensemble classificateur pour la classification de données dynamiques. Application à un problème de qualité d'air intérieur. Journal Européen des Systèmes Automatisés, 49(3), 375-391 (2016).

[6] Thomas P., Derigent W., Suhner M.C.: Prediction model adaptation thanks to control chart monitoring. Application to pollutants prediction. 6th International Joint Conference on Computational Intelligence IJCCI'14, Rome, Italie, 22 – 24 october (2014).

[7] Noyel M., Thomas P., Thomas A., Charpentier P.: Reconfiguration process for neuronal classification models: Application to a quality monitoring problem. Computers in Industry, 83, 78-91 (2016).

[8] Rumelhart D.E., McClelland J.L.: Parallel distributed processing, The MIT Press, Cambridge, England (1986).

[9] Levenberg K.: A method for the solution of certain nonlinear problem in least squares. Quart. Appl. Math., 2:164-168 (1944).

[10] Marquardt D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Appl. Math., 11:431-441 (1963).

[11] Schmidt, W.F., Kraaijveld, M.A., Duin, R.P.W.: Feedforward neural networks with random weights, Proc. of 11th IAPR Int. Conference on Pattern Recognition Methodology and Systems, 2, 1-4 (1992).

[12] Xie L., Yang Y., Zhou Z., Zheng J., Tao M., Man Z.: Dynamic neural modeling of fatigue crack growth process in ductile alloys. Information Sciences, 364–365, 167-183 (2016).

[13] Ye H., Cao F., Wang D., Li H.: Building feedforward neural networks with random weights for large scale datasets. Expert Systems with Applications, 106, 233-243 (2018).

[14] WHO World Health Organization: WHO guidelines for indoor air quality: selected pollutants. Copenhagen: WHO Regional Office for Europe, (2010).

[15] ISO 16000-6: Indoor air - Part 6: determination of volatile organic compounds in indoor air and test chamber air by active sampling on Tenax TA sorbent, thermal desorption and gas chromatography using MS or MS/FID. Berlin: Beuth Verlag, (2011).

[16] Jones A.P.: Indoor air quality and health. Atmospheric Environment. 33, 4535-4564 (1999).

[17] ATMO: http://www.atmo-grandest.eu/ (2018).

[18] Challoner A., Pilla F., Gill L.: Prediction of indoor air exposure from outdoor air quality using an artificial neural network model for inner city commercial buildings. International Journal of Environmental Research and Public Health, 12(12), 15233-15253 (2015).

[19] Schieweck A., Uhde E., Salthammer T., Salthammer L.C., Morawska L, Mazaheri M., Kumar P.: Smart homes and the control of indoor air quality. Renewable and Sustainable Energy Reviews, 94, 705-718 (2018).

[20] Salamone F., Belussi L., Danza L., Galanos T., Ghellere M., Meroni I.: Design and development of a nearable wireless system to control indoor air quality and indoor lighting quality. Sensors, 17(5), 1021 (2017).

[21] Tijani K., Ngo D.Q., Ploix S., Haas B., Dugdale J.: Dynamic Bayesian networks to simulate occupant behaviours in office buildings related to indoor air quality. 14th International Conference of the International Building Performance Simulation Association (IBPSA) Hyderabad, India 7 - 9 december (2015).

[22] Xie H.X.H., Ma F.M.F., Bai Q.B.Q.: Prediction of indoor air quality using artificial neural networks. 5th International Conference on Natural Computation, Tianjin, China, 14-16 August 2, 414–418 (2009).

[23] Thomas P., Derigent W., Suhner M.C.: Un ensemble classificateur pour la classification de données dynamiques. Application à un problème de qualité d'air intérieur. 10ième Conférence international de Modélisation et Simulation MOSIM'14, Nancy, France, 5 – 7 november (2014).

[24] Saad S.M., Andrew A.M., Shakaff A.Y., Saad A.R.M., Kamarudin A.M.Y., Zakaria A.: Classifying sources influencing indoor air quality (IAQ) using artificial neural network (ANN). Sensors, 15, 11665-11684; doi:10.3390/s150511665 (2015).

[25] Abd Rahman,N.H., Lee M.H., Latif M.T., Suhartono S.: Forecasting of air pollution index with artificial neural network. Jurnal Teknologi, 63(2), 59–64 (2013).

[26] Yu T.C., Lin C.C. :An intelligent wireless sensing and control system to improve indoor air quality: monitoring, prediction, and preaction. International Journal of Distributed Sensor Networks (2015).

[27] Thomas, P., Bloch, G., Sirou, F., Eustache, V.: Neural modeling of an induction furnace using robust learning criteria. Journal of Integrated Computer Aided Engineering, 6, 1, 5-23 (1999).

[28] Shewhart, W.A.: Economic Control of Quality of Manufactured Product, Van Nostrand Reinhold Company, Inc.: Princeton, NJ. (1931).

[29] NIST/SEMATECH, e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/ (2012).

[30] Moore, E. H.: On the reciprocal of the general algebraic matrix. Bulletin of the American Mathematical Society. 26 (9), (1920).

[31] Penrose, R.: A generalized inverse for matrices. Proceedings of the Cambridge Philosophical Society. 51, 406–13. (1955)

[32] Huang, G.B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE trans. on Neural Networks, 17(4), 879-892 (2006).

[33] Thomas P., Suhner M.C.: A new multilayer perceptron pruning algorithm for classification and regression applications. Neural Processing Letters. 42(2), 437-458 (2015).