Incremental-based Extreme Learning Machine algorithms for Time-Variant Neural Networks

# Incremental-based Extreme Learning Machine algorithms for Time-Variant Neural Networks

Yibin Ye, Stefano Squartini, and Francesco Piazza

A3LAB, Department of Biomedics, Electronics and Telecommunications,
Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
{yibin.ye,s.squartini,f.piazza}@univpm.it
http://www.a3lab.dibet.univpm.it

**Abstract.** Extreme Learning Machine (ELM) is a novel learning algorithm for Neural Networks (NN) much faster than the traditional gradient-based learning techniques, and many variants, extensions and applications in the NN field have been appeared in the recent literature. Among them, an ELM approach has been applied to training Time-Variant Neural Networks (TV-NN), with the main objective to reduce the training time. Moreover, interesting approaches have been proposed to automatically determine the number of hidden nodes, which represents one of the limitations of original ELM algorithm for NN. In this paper, we extend the Error Minimized Extreme Learning Machine (EM-ELM) algorithm along with other two incremental based ELM methods to the time-variant case study, which is actually missing in the related literature. Comparative simulation results show the the proposed EM-ELM-TV is efficient to optimally determine the basic network architecture guaranteeing good generalization performances at the same time.

## 1 Introduction

A fast learning algorithm called Extreme Learning Machine (ELM) introduced by Huang et al[1, 2], has recently caught much attention within the Neural Network (NN) research community. ELM is designed for single hidden layer feed-forward neural networks (SLFNs): it randomly chooses hidden nodes and then determines the output weight analytically. However, one of the open problem of ELM is how to assign the number of hidden neurons, which is the only factor that needs to be set by users, usually by trial-and-error. The first hidden nodes increment algorithm for ELM, referred to as Incremental Extreme Learning Machine (I-ELM)[3], randomly adds nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added. However, due to the random generation of added hidden nodes, some of which may play a very minor role in the network output and thus may eventually increase the network complexity. In order to avoid this issue, an Enhanced method for I-ELM, called EI-ELM[4] have been proposed. At each learning step, several hidden nodes are randomly generated and the one leading to least error is then selected and added to the network.

Recently, another ELM-based hidden nodes incremental learning algorithm referred to as Error Minimized Extreme Learning Machine (EM-ELM)[5] was proposed. It can also add random hidden nodes to SLFNs one by one or even group by group, with all the previous output weights updated accordingly at each step. Compared with I-ELM and EI-ELM which keep the output weights of existing hidden nodes fixed when adding a new hidden node, EM-ELM attain a much better generalization performance.

Time-Variant Neural Networks(TV-NN) represent a relevant example in the field of neural architectures working properly in non-stationary environments. Such networks have time-variant weights, each being a linear combination of a certain set of basis functions. Titti et al[6], proposed an extended version of the Back-Propagation(BP) algorithm to suitably train such networks. An Extreme Learning Machine approach is also developed for TV-NN(ELM-TV-NN), introduced by Cingolani et al[7], accelerating the training procedure significantly. However, ELM-TV-NN also has the problem to determine the size of network, e.g. the number of hidden nodes, as ELM does.

In this paper, we attempt to extend I-ELM, EI-ELM and EM-ELM to time-variant networks, as well as testing and comparing them in 3 different task scenarios. We will show that similar behavior to the one achievable in the time-invariant case can be obtained, proving the effectiveness of the algorithm generalization to the TV-NN case study.

Due to the space constraint, we do not present more details for ELM algorithm and its three related incremental-based variants I-ELM, EI-ELM and EM-ELM. Please refer to [2–5] for more information.

## 2 The ELM Approach for Time-Variant Neural Networks

The application and extension of ELM to the time-variant case has been studied in [7]. In a time-variant neural network, the input weights, or output weights, or both are changing through the training and testing time. Each weight can be expressed as a linear combination of a certain set of basis functions: $w[n] = \sum_{b=1}^{B} f_b[n] \cdot w_b$, in which $f_b[n]$ is the known orthogonal function at $n$-th time instant of $b$-th order, $w_b$ is the $b$-th order coefficient of the basic function to construct time-variant weight $w_n$, while $B$ is the total number of the bases preset by user.

If time-variant input weights are introduced in a SLFN, hidden neuron output function can be rewritten as: $h_k[n] = g\left(\sum_{i=0}^{I} x_i[n] \cdot w_{ik}[n]\right)$, where $w_{ik}[n] = \sum_{b=1}^{B} f_b[n] \cdot w_{b,ik}$. Similarly, if time-variant output weights are introduced, the standard output equation can be rewritten as: $\sum_{k=1}^{K} h_k[n] \cdot \beta_{kl}[n] = t_l[n]$, where $\beta_{kl}[n] = \sum_{b=1}^{B} f_b[n] \cdot \beta_{b,kl}$.

To train a TV-NN, $\{w_{b,ik}\}$ can be randomly generated and then hidden-layer output matrix $\mathbf{H}$ can be computed. However, the values of a set of output weight parameters $\{\beta_{b,kl}\}$ can not be so straightforward calculated. Some transformations are needed. Let us expand the time-variant output weights

$\beta_{kl}[n]$ and assume the following notation: $\mathbf{f}[n] = [f_1[n], f_2[n], \ldots, f_B[n]]^T \in \mathbb{R}^B$, $\mathbf{h}[n] = [h_1[n], h_2[n], \ldots, h_K[n]]^T \in \mathbb{R}^K$, $\boldsymbol{\beta}_{kl} = [\beta_{1,kl}, \beta_{2,kl}, \ldots, \beta_{B,kl}]^T \in \mathbb{R}^B$, $\boldsymbol{\beta}_{(l)} = [\boldsymbol{\beta}_{1l}, \boldsymbol{\beta}_{2l}, \ldots, \boldsymbol{\beta}_{Kl}] \in \mathbb{R}^{B\times K}$, $\boldsymbol{\omega}_{(l)} = [\boldsymbol{\beta}_{1l}^T, \boldsymbol{\beta}_{2l}^T, \ldots, \boldsymbol{\beta}_{Kl}^T]^T \in \mathbb{R}^{B\cdot K\times 1}$. We can now state:

$$\begin{aligned} t_l[n] &= \sum_{k=1}^{K} h_k[n] \cdot \left( \sum_{b=1}^{B} f_b[n] \cdot \beta_{b,kl} \right) = \sum_{k=1}^{K} \mathbf{f}[n]^T \cdot \boldsymbol{\beta}_{kl} \cdot h_k[n] \\ &= \mathbf{f}[n]^T \cdot \boldsymbol{\beta}_{(l)} \cdot \mathbf{h}[n] = \left( \mathbf{h}[n]^T \otimes \mathbf{f}[n]^T \right) \cdot \boldsymbol{\omega}_{(l)} \end{aligned} \tag{1}$$

where $\otimes$ denotes the *Kronecker product* of $\mathbf{h}[n]^T$ and $\mathbf{f}[n]^T$. The last step consists in: $vec(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})vec(\mathbf{B})$, (note that $vec(t_l[n]) = t_l[n]$.) and $\boldsymbol{\omega}_{(l)} = vec(\boldsymbol{\beta}_{(l)})$ is the vectorization of the matrix $\boldsymbol{\beta}_{(l)}$ formed by stacking the columns of $\boldsymbol{\beta}_{(l)}$ into a single column vector.

Moreover, we define: $\mathbf{G} = \mathbf{H} * \mathbf{F}$, where $\mathbf{H} = [\mathbf{h}[1], \mathbf{h}[2], \ldots, \mathbf{h}[N]]^T = \{h_k[n]\} \in \mathbb{R}^{N\times K}$, $\mathbf{F} = [\mathbf{f}[1], \mathbf{f}[2], \ldots, \mathbf{f}[N]]^T = \{f_b[n]\} \in \mathbb{R}^{N\times B}$, $*$ denotes the *Khatri-Rao product* of matrices $\mathbf{H}$ and $\mathbf{F}$, with $\mathbf{h}[n]^T$ and $\mathbf{f}[n]^T$ as their submatrices, respectively. Further assuming that: $\mathbf{T} = \{t_l[n]\} \in \mathbb{R}^{N\times L}$, $\boldsymbol{\Omega} = [\boldsymbol{\omega}_{(1)}, \boldsymbol{\omega}_{(2)}, \ldots, \boldsymbol{\omega}_{(L)}] \in \mathbb{R}^{B\cdot K\times L}$, we get: $\mathbf{G} \cdot \boldsymbol{\Omega} = \mathbf{T}$

Since $\mathbf{F}$ is obtained by the type of the basis function predetermined by the user; $\mathbf{H}$ can be calculated once input weight parameters are randomly generated. Hence we can get $\mathbf{G}$. Similar to the ELM algorithm described in previous section, the time-variant output weight matrix $\boldsymbol{\Omega}$ can be computed by:

$$\hat{\boldsymbol{\Omega}} = \mathbf{G}^{\dagger} \cdot \mathbf{T} \tag{2}$$

where $\mathbf{G}^{\dagger}$ is the MP inverse of matrix $\mathbf{G}$, and consequently, $\hat{\boldsymbol{\Omega}}$ is a set of optimal output weight parameters minimizing the training error.

## 3 Proposed Incremental-based ELM algorithms for TV-NN

### 3.1 I-ELM-TV and EI-ELM-TV

It is proved in [3] and [4] that for one specific output neuron, when the $k$-th hidden neuron is added and $\beta_k = \frac{\tilde{\mathbf{h}}_k^T \cdot \tilde{\mathbf{e}}_{k-1}}{\tilde{\mathbf{h}}_k^T \cdot \tilde{\mathbf{h}}_k}$, $\|\tilde{\mathbf{e}}_k\| = \|\tilde{\mathbf{t}} - (\tilde{\mathbf{t}}_{k-1} + \beta_k \tilde{\mathbf{h}}_k)\|$ achieves its minimum and the sequence $\{\|\tilde{\mathbf{e}}_k\|\}$ decreases and converges. Note that $\beta_k = \frac{\tilde{\mathbf{h}}_k^T \cdot \tilde{\mathbf{e}}_{k-1}}{\tilde{\mathbf{h}}_k^T \cdot \tilde{\mathbf{h}}_k}$, is a special case of $\beta_k = \tilde{\mathbf{h}}_k^{\dagger} \tilde{\mathbf{e}}_{k-1}$ when $\tilde{\mathbf{e}}_{k-1}$ and $\tilde{\mathbf{h}}_k$ are vectors. Actually, the MP generalized inverse of $\tilde{\mathbf{h}}_k$ is just $\tilde{\mathbf{h}}_k^{\dagger} = (\tilde{\mathbf{h}}_k^T \cdot \tilde{\mathbf{h}}_k)^{-1} \tilde{\mathbf{h}}_k^T$ For our time-variant case, this can also be extended to matrix computations. Let $\delta\boldsymbol{\Omega}_k = \begin{bmatrix} \beta_{1,k1}, \cdots, \beta_{1,kL} \\ \cdots, \cdots, \cdots \\ \beta_{B,k1}, \cdots, \beta_{B,kL} \end{bmatrix}_{B\times L}$ and $\delta\mathbf{G}_k = \begin{bmatrix} h_k[1] \cdot \mathbf{f}[1]^T \\ \cdots \\ h_k[N] \cdot \mathbf{f}[N]^T \end{bmatrix}_{N\times B}$, (Note $\boldsymbol{\Omega} = \begin{bmatrix} \delta\boldsymbol{\Omega}_1 \\ \cdots \\ \delta\boldsymbol{\Omega}_K \end{bmatrix}$ and $\mathbf{G} = [\delta\mathbf{G}_1, \cdots, \delta\mathbf{G}_K]$.) Similarly, if $\delta\boldsymbol{\Omega}_k = \delta\mathbf{G}_k^{\dagger} \cdot \mathbf{E}_{k-1}$,

$\|\mathbf{E}_k\| = \|\mathbf{T} - (\mathbf{T}_{k-1} + \delta\mathbf{G}_k\delta\mathbf{\Omega}_k)\|$ achieve its minimum and the sequence $\{\|\mathbf{E}_k\|\}$ would decrease and converges.

It is noted in [4] that some newly added hidden nodes may make residual error reduce less than others. In the Enhanced I-ELM method, at any step, among $P$ trial of hidden nodes, the hidden nodes with greatest residual error reduction is chosen and added. This method is also applied in this paper.

Hence, we have our time-variant version of I-ELM and EI-ELM. Assume we have a set of training data $\{(\mathbf{x}[n], \mathbf{t}[n])\}_{n=1}^{N}$, the target output matrix $\mathbf{T}$, the residual matrix $\mathbf{E}$, the maximum number of hidden nodes $K_{max}$, and the expected learning accuracy $\epsilon$. We get Algorithm 1.

**Algorithm 1** I-ELM-TV(in the case of $P = 1$) and EI-ELM-TV

Let $k = 0$ and residual error $\mathbf{E} = \mathbf{T}$,
**while** $k < K_{max}$ and $\|\mathbf{E}\| > \epsilon$, **do**
  Increase by one the number of hidden nodes: $k = k + 1$;
  **for** $p = 1 : P$ **do**
    Assign random input weight and bias $\{w_{(p)b,ik}\}$ for new hidden node;
    Calculate the hidden layer output submatrix for new hidden node

$$\delta\mathbf{G}_{(p)} = \begin{bmatrix} h_{(p)k}[1] \cdot \mathbf{f}[1]^T \\ \cdots \\ h_{(p)k}[N] \cdot \mathbf{f}[N]^T \end{bmatrix}_{N \times B}$$

    Calculate the output weight $\delta\mathbf{\Omega}_{(p)}$ for the new hidden node

$$\delta\mathbf{\Omega}_{(p)} = \delta\mathbf{G}_{(p)}^{\dagger} \cdot \mathbf{E}$$

    Calculate the residual error after adding the new hidden node $k$:

$$\mathbf{E}_{(p)} = \mathbf{E} - \delta\mathbf{G}_{(p)} \cdot \delta\mathbf{\Omega}_{(p)}$$

  **end for**
  Let $p^* = \{p | min_{1 \leqslant p \leqslant P} \|\mathbf{E}_{(p)}\|\}$.
  Set $\mathbf{E} = \mathbf{E}_{(p^*)}$, $\{w_{b,ik} = w_{(p^*)b,ik}\}$, and $\delta\mathbf{\Omega}_k = \delta\mathbf{\Omega}_{(p^*)}$.
**end while**
The output weight matrix would be $\mathbf{\Omega} = \begin{bmatrix} \delta\mathbf{\Omega}_1 \\ \cdots \\ \delta\mathbf{\Omega}_K \end{bmatrix}_{B \cdot K \times L}$

### 3.2 EM-ELM-TV

Similarly, with certain transformation, Error Minimized ELM approach can also be extended in time-variant case. Replace $\mathbf{H}_0$, $\delta\mathbf{H}_k$, $\beta$ with $\mathbf{G}_1$,$\delta\mathbf{G}_k$, $\mathbf{\Omega}$, respectively, we get our time-variant version of EM-ELM. For the sake of presentation clarity, we only add hidden nodes one by one in our proposed EM-ELM-TV

algorithm, which can be easily generalized to the group-by-group node addition means.

Assume we have a set of training data $\{(\mathbf{x}[n], \mathbf{t}[n])\}_{n=1}^{N}$, the target matrix $\mathbf{T}$, the residual matrix $\mathbf{E}_k = \mathbf{G}_k\mathbf{G}_k^{\dagger}\mathbf{T} - \mathbf{T}$, the maximum number of hidden nodes $K_{max}$, and the expected learning accuracy $\epsilon$. We get Algorithm 2.

**Algorithm 2** EM-ELM-TV

1: Randomly generate one hidden node.
2: Calculate the time-variant hidden layer output matrix $\mathbf{G}_1$ by (**??**) and (**??**):

$$\mathbf{G}_1 = \begin{bmatrix} h_1[1] \cdot \mathbf{f}[1]^T \\ \cdots \\ h_1[N] \cdot \mathbf{f}[N]^T \end{bmatrix}_{N \times B}$$

3: Calculate the output error $\|\mathbf{E}_1\| = \|\mathbf{G}_1\mathbf{G}_1^{\dagger}\mathbf{T} - \mathbf{T}\|$.
4: Let $k = 1$
5: **while** $k < K_{max}$ and $\|\mathbf{E}_k\| > \epsilon$, **do**
6: Randomly add another hidden node. The corresponding hidden layer output matrix becomes $\mathbf{G}_{k+1} = [\mathbf{G}_k, \delta\mathbf{G}_k]$, where

$$\delta\mathbf{G}_k = \begin{bmatrix} h_k[1] \cdot \mathbf{f}[1]^T \\ \cdots \\ h_k[N] \cdot \mathbf{f}[N]^T \end{bmatrix}_{N \times B}$$

7: Update the output weight $\mathbf{\Omega}$

$$\mathbf{D}_k = ((\mathbf{I} - \mathbf{G}_k\mathbf{G}_k^{\dagger})\delta\mathbf{G}_k)^{\dagger}$$
$$\mathbf{U}_k = \mathbf{G}_k^{\dagger} - \mathbf{G}_k^{\dagger}\delta\mathbf{G}_k\mathbf{D}_k$$
$$\mathbf{\Omega}^{(k+1)} = \mathbf{G}_{k+1}^{\dagger}\mathbf{T} = \begin{bmatrix} \mathbf{U}_k \\ \mathbf{D}_k \end{bmatrix}\mathbf{T}$$

8: $k = k + 1$
9: **end while**

## 4 Simulation Results

In this section, we compare I-ELM-TV, EI-ELM-TV and EM-ELM-TV with ELM-TV in three identification problems. All the data depicted in figures are the average values of 10 trials. All the programs are run in MATLAB 7.8.0 environment with Windows Vista, Intel Core2 Duo CPU P8400 2.26GHz. The number $P$ of trials of assigning random hidden nodes at each step for EI-ELM-TV, is set to 10. The chosen basis function type for all these time-variant algorithms is Prolate [6] and the number of bases have been set to 3. Varying the basis function type and number leads to different performances, but similar comparative

conclusions as those drawn in following subsections can be done: therefore, for the sake of conciseness, such results have not been reported here.

### 4.1 Time-variant perceptron system identification

In the first case, we construct a simple time-variant system: $y[n] = g(w[n] \cdot x[n])$ , where $g(x)$ is a sigmoid activation function $g(x) = \frac{1}{1+e^{-x}}$, and $w[n]$ is a single time-variant coefficient, which is combination of 3 prolate function bases, $w[n] = \sum_{b=1}^{3} f_b[n] \cdot w_b$, with setting $w_b = 0.5, (b = 1, 2, 3)$. The inputs are normalized (within the range $[-1, 1]$).

In the simulation of single perceptron system, hidden neurons are gradually increase one by one from 1 to 20. Results show that EM-ELM-TV has similar performance as ELM-TV in terms of testing accuracy and outperforms over I-ELM-TV and EI-ELM-TV, but reach its minimum accuracy by only about 17 hidden nodes. Though the training time of EM-ELM-TV is more than ELM-TV when the number of hidden nodes is small, it only increase slightly when more hidden nodes are added. On the other hand, EI-ELM-TV achieves better testing accuracy than I-ELM-TV as expected, at the cost of more training time. Therefore, the incremental ELM algorithms seem not to be effective w.r.t. the common ELM. It can be justified saying that the simplicity of the addressed task do not ask for enough hidden nodes to appreciate the effectiveness of the EM based approach, as it will occur in next simulations.

### 4.2 Time-variant MLP system identification

The system to be identified here is the same with that in [6] and [7]: a time-variant IIR-buffered MLP with 11 input lines, one 5 neurons hidden layer, one neuron output layer. The input weights and output weights are combination of 3 Chebyshev basis functions; the lengths of the input and output TDLs are equal to 6 and 5 respectively. Note that the output neuron of this system is not linear, both the hidden neurons and output neuron use tangent sigmoid activation function.

A wider range of hidden nodes from 1 to 200 are tested in this scenario. The accuracy performance of EM-ELM-TV is quite the same with that of ELM-TV, and as the number of hidden nodes becomes larger, the training time of EM-ELM-TV is much less than that of ELM-TV.

### 4.3 Time-Variant Narendra system identification

The next test identification system is a modified one addressed in [8], adding the coefficients $a[n]$ and $b[n]$ to form a time-variant system, as done in [6] and [7] :

$$y[n] = a[n] \cdot \frac{y[n-1] \cdot y[n-2] \cdot y[n-3] \cdot x[n-1] \cdot (y[n-3]-1) + x[n]}{1 + b[n] \cdot (y[n-3]^2 + y[n-2]^2)} \quad (3)$$

Simulation results for Narendra system are depicted in Fig 1 and Fig 2. For I-ELM-TV and EI-ELM-TV, better testing accuracies can be achieved gradually and smoothly as new hidden nodes are added. However, this is not true for ELM-TV and EM-ELM-TV. When adding new hidden nodes to EM-ELM-TV, the training RMSE does decrease, but the testing RMSE may not. We can conclude from these that I-ELM-TV and EI-ELM-TV produce more stable results than ELM-TV and EM-ELM-TV in Narendra system, though they converge much slower. Again, same behavior as in the time-invariant case is recognized.

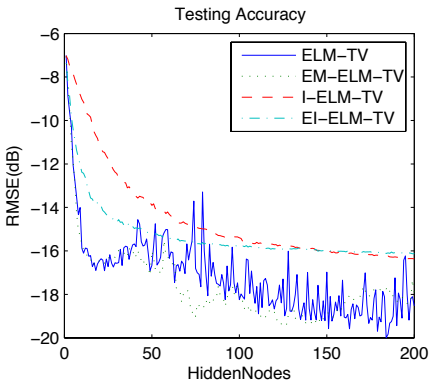


**Fig. 1.** Testing Accuracy of 4 algorithms for Narendra System

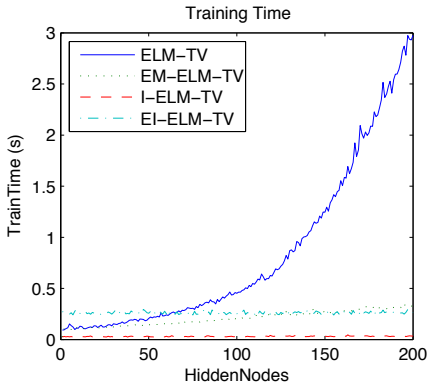


**Fig. 2.** Training Time of 4 algorithms for Narendra System

The performance comparison in terms of necessary hidden nodes between ELM-TV and EM-ELM-TV, has also been conducted in these time-variant systems. The target training RMSE(dB) $\epsilon$ are set as: -40 for perceptron system, -35 for MLP system and -18 for Narendra system. The optimal number of hidden nodes for ELM-TV is obtained by trial-and-error. Table 1 displays performance evaluation between ELM-TV and EM-ELM-TV, since I-ELM-TV and EM-ELM-TV are not able to reach the training goals of them within 500 nodes. Focusing on the MLP and Narendra Systems case studies for reasons explained above, results reported in Table 1 prove that EM-ELM-TV has similar generalization performance and optimal number of hidden nodes attainable with ELM-TV, but at reduced training time. Therefore, EM-ELM-TV is able to optimally select the number of hidden nodes more efficiently than the trial-and-error approach typically used in common ELM-TV, as well as concluded in the time-invariant case.

## 5 Conclusions

In this paper, we have proposed three incremental-based Extreme Learning Machine algorithms for Time-Variant Neural Networks training. They are the extensions of the corresponding time-invariant counterparts and they have been

**Table 1.** Performance Comparisons for the Number(average of 10 trials) of Hidden Nodes When the Expected Accuracy is Reached

| Systems [stop RMSE(dB)] | Algorithms | Training Time (s) | Testing RMSE(dB) | Hidden Nodes |
|---|---|---|---|---|
| Perceptron [-40] | ELM-TV | 0.0775 | -41.80 | 13.7 |
| | EM-ELM-TV | 0.1166 | -41.75 | 13.2 |
| MLP [-35] | ELM-TV | 3.0707 | -30.69 | 197.7 |
| | EM-ELM-TV | 0.3463 | -30.07 | 203.8 |
| Narendra [-18] | ELM-TV | 0.2039 | -16.13 | 48.1 |
| | EM-ELM-TV | 0.1604 | -16.75 | 49.5 |

addressed as I-ELM-TV, EI-ELM-TV and EM-ELM-TV. The main advantage of the incremental approach consists in automatically determining the number of hidden nodes, avoiding to use the trial-and-error method typically adopted in standard ELM. As it occurs in the time-invariant case, the EM method, in contrast to the others, is able to get generalization performances comparable to the original ELM, also significantly reducing the training time (when large number of hidden nodes are required). Future works are intended to apply the promising EM paradigm to the optimal selection of the number of bases functions, which is still assign by the users before TV-NN training.

## References

1. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proc. IEEE International Joint Conference on Neural Networks. Volume 2. (July 25–29, 2004) 985–990
2. Huang, G.B., Zhu, Q.Y., Siew., C.K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1-3) (2006) 489 – 501
3. Huang, G., Chen, L., Siew, C.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks **17**(4) (2006) 879
4. Huang, G., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16-18) (2008) 3460–3468
5. Feng, G., Huang, G.B., Lin, Q., Gay, R.: Error minimized extreme learning machine with growth of hidden nodes and incremental learning. **20**(8) (August 2009) 1352–1357
6. Titti, A., Squartini, S., Piazza, F.: A new time-variant neural based approach for nonstationary and non-linear system identification. In: Proc. IEEE International Symposium on Circuits and Systems ISCAS 2005. (May 23–26, 2005) 5134–5137
7. Cingolani, C., Squartini, S., Piazza, F.: An extreme learning machine approach for training time variant neural networks. In: Proc. IEEE Asia Pacific Conference on Circuits and Systems APCCAS 2008. (November 2008) 384–387
8. Narendra, K., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. Neural Networks, IEEE Transactions on **1**(1) (mar 1990) 4 –27