ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Random Search Enhancement of Error Minimized Extreme Learning Machine

Yuan Lan[1], Yeng Chai Soh[2] and Guang-Bin Huang[3]

1, 2, 3 - Nanyang Technological University - School of Electrical and Electronic Engineering - {lany0001, eycsoh, egbhuang}@ntu.edu.sg - Singapore

Abstract. Error Minimized Extreme Learning Machine (EM-ELM) proposed by Feng et al. [1] can automatically determine the number of hidden nodes in generalized Single-hidden Layer Feedforward Networks (SLFNs). We recently found that some of the hidden nodes that are added into the network may play a very minor role in the network output, which increases the network complexity. Hence, this paper proposes an Enhancement of EM-ELM (referred to as EEM-ELM), which introduce a selection phase based on the random search method. The empirical study shows that EEM-ELM leads to a more compact network structure.

1 Introduction

Recently, Feng et al [1] provided a incremental approach referred to as Error Minimized Extreme Learning Machine (EM-ELM) based on the Extreme Learning Machine (ELM) [2, 3, 4, 5]. EM-ELM is a simple and efficient incremental learning algorithm for generalized SLFNs and can determine the number of hidden nodes automatically that may not be neural alike. It adds random hidden nodes to SLFNs one-by-one or group-by-group with fixed or varying group size. Then the output weights of the network are updated incrementally during the network growth, which dramatically reduces the computational complexity. It has been proved that EM-ELM is convergent in theory. However, we recently investigated that some of the hidden nodes in EM-ELM network may play a very minor role in the network output, which may increase the network complexity. And we realize that there is not any selection of the hidden nodes before they are added to the EM-ELM network.

In this paper, we propose an Enhancement of EM-ELM (referred to as EEM-ELM), which introduces a selection phase based on the random search method. At each incremental learning step, several hidden nodes are randomly generated and the hidden node that leads to highest residual error reduction will be added into the network, and then the output weights are updated incrementally in the same way of original EM-ELM. The empirical study shows that EEM-ELM leads to a more compact network structure.

2 Review of EM-ELM

EM-ELM is an error minimization based method in which the number of hidden nodes can grow one-by-one or group-by-group until optimal. The approach

can significantly reduce the computational complexity and its convergence was proved as well. In this paper, we only consider the one-by-one incremental mode.

Assume we have a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N}$, the maximum number of hidden nodes L_{max}, and the expected learning accuracy ϵ. There are two phases in EM-ELM algorithm.

Initialization Phase

1. Randomly generate a hidden node $(\mathbf{a}_1, b_1)$, where $\mathbf{a}$ and b are the parameters of hidden node. And set $j = 0$ and $L_0 = 1$.

2. Calculating the hidden layer output matrix: $\mathbf{h}_0$

$$\mathbf{h}_0 = \left[\begin{array}{ccc} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_1, b_1, \mathbf{x}_N) \end{array} \right]^T \tag{1}$$

   where $G(\mathbf{x})$ is the activation function and $G(\mathbf{a}_1, b_1, \mathbf{x}_i)$ denotes the output of the hidden node w.r.t input $\mathbf{x}_i$.

3. Calculating the corresponding output error $E(\mathbf{h}_0) = \|\mathbf{h}_0\mathbf{h}_0^{\dagger}\mathbf{t} - \mathbf{t}\|$.

Recursively Growing Phase:
while $L_j < L_{max}$ and $E(\mathbf{H}_j) > \epsilon$

1. Randomly add a hidden node to the existing SLFNs. The number of hidden nodes $L_{j+1} = L_j + 1$ and the corresponding hidden layer output matrix $\mathbf{H}_{j+1} = [\mathbf{H}_j \quad \delta\mathbf{h}_j]$, where $\delta\mathbf{h}_j$ is shown below.

$$\delta\mathbf{h}_j = \left[\begin{array}{ccc} G(\mathbf{a}_{L_j+1}, b_{L_j+1}, \mathbf{x}_1) & \cdots & G(\mathbf{a}_{L_j+1}, b_{L_j+1}, \mathbf{x}_N) \end{array} \right]^T \tag{2}$$

2. Updating the output weight β [6]

$$\begin{aligned} \mathbf{D}_j &= \frac{\delta\mathbf{h}_j^T(\mathbf{I} - \mathbf{H}_j\mathbf{H}_j^{\dagger})}{\delta\mathbf{h}_j^T(\mathbf{I} - \mathbf{H}_j\mathbf{H}_j^{\dagger})\delta\mathbf{h}_j} \\ \mathbf{U}_j &= \mathbf{H}_j^{\dagger} - \mathbf{H}_j^{\dagger}\delta\mathbf{h}_j\mathbf{D}_j \\ \beta^{(j+1)} &= \mathbf{H}_{j+1}^{\dagger}\mathbf{t} = \left[\begin{array}{c} \mathbf{U}_j \\ \mathbf{D}_j \end{array} \right]\mathbf{t} \end{aligned} \tag{3}$$

3. $j = j + 1$.

endwhile

3 Proposed enhancement of EM-ELM (EEM-ELM)

In EM-ELM, the hidden nodes are randomly generated and added to the network sequentially. During the study of EM-ELM, we investigate that some newly added hidden nodes may be more efficient in reducing the residual error as

compared to other hidden nodes. Hence, we propose an Enhancement of EM-ELM (referred to as EEM-ELM) by applying random search method. In EEM-ELM, the hidden node is added to the network one-by-one. At each incremental learning step, k hidden nodes are randomly generated and the hidden node that leads to highest residual error reduction will be added to the network, and then the output weights are updated incrementally in the same way of original EM-ELM. The EEM-ELM can be summarized as follows:

EEM-ELM Algorithm: Given a a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N}$, activation function $G(x)$, maximum number of hidden nodes L_{max} allowed to add into the network, number of hidden nodes k for selection in each step, and the expected learning accuracy ϵ.

Initialization step: Let the number of hidden nodes in the network $L = 0$, step $j = 0$ and the initial residual error is equal to the target $\mathbf{t}$.

Incremental learning step:

while $L_j < L_{max}$ and $E(\mathbf{H}_j) > \epsilon$

- Let $L_{j+1} = L_j + 1$.
- **for** $i = 1 : k$
  1. Generate one random hidden node $(\mathbf{a}^i, b^i)$ and add it to the existing network, where $\mathbf{a}^i$ and b^i are parameters of the ith hidden node that is generated for selection. The number of hidden nodes is L_{j+1}, and the corresponding hidden layer output matrix $\mathbf{H}_{j+1}^i = [\mathbf{H}_j \quad \delta\mathbf{h}_j^i]$, where $\delta\mathbf{h}_j^i = \left[\begin{array}{ccc} G(\mathbf{a}_{L_j+1}^i, b_{L_j+1}^i, \mathbf{x}_1) & \cdots & G(\mathbf{a}_{L_j+1}^i, b_{L_j+1}^i, \mathbf{x}_N) \end{array}\right]^T$.
  2. Update the output weight β_i

$$\begin{aligned} \mathbf{D}_j &= \frac{\delta\mathbf{h}_j^{iT}(\mathbf{I} - \mathbf{H}_j\mathbf{H}_j^\dagger)}{\delta\mathbf{h}_j^{iT}(\mathbf{I} - \mathbf{H}_j\mathbf{H}_j^\dagger)\delta\mathbf{h}_j^i} \\ \mathbf{U}_j &= \mathbf{H}_j^\dagger - \mathbf{H}_j^\dagger\delta\mathbf{h}_j^i\mathbf{D}_j \\ \beta_i^{(j+1)} &= \mathbf{H}_{j+1}^{i\dagger}\mathbf{t} = \left[\begin{array}{c} \mathbf{U}_j \\ \mathbf{D}_j \end{array}\right]\mathbf{t} \end{aligned} \tag{4}$$

  3. Calculate the corresponding output error $E_i(\mathbf{H}_{j+1}^i) = \|\mathbf{H}_{j+1}^i\beta_i^{(j+1)} - \mathbf{t}\|$.

  endfor

- Let $i^* = \{i | \min_{1\le i\le k} \|E_i\|\}$. Hence, hidden node $(\mathbf{a}^{i^*}, b^{i^*})$ leads to highest error reduction and is added to the network. And then set $E = E_{i^*}$, $\mathbf{a}_{L_j+1} = \mathbf{a}^{i^*}$, $b_{L_j+1} = b^{i^*}$, and $\beta^{(j+1)} = \beta_{i^*}$.
- $j = j + 1$.

endwhile

ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

4 Performance Evaluation of EEM-ELM

In this section, the performance of EEM-ELM is evaluated on the benchmark problems described in Table 1, which includes six applications [7]. We compare the results of EEM-ELM with EM-ELM [1], ELM [2] and OP-ELM [8, 9] algorithms.

Table 1: Specification of Benchmark Datasets

Datasets	# Attributes	#Classes	# Training Data	# Testing Data
Abalone	8	-	2000	2177
Auto-MPG	7	-	200	198
Boston Housing	13	-	250	256
Machine CPU	6	-	100	109
Image Segmentation	19	7	1210	1100
Satellite Image	36	6	3217	3218

In our simulations, data normalization has been done the same as in [1]. Twenty trials of simulations were conducted for each applications with EEM-ELM, EM-ELM, ELM and OP-ELM algorithms. The activation function used in the comparison was the sigmoidal additive activation function $G(\mathbf{a}, b, \mathbf{x}) = (1)/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$ for EEM-ELM, EM-ELM and ELM. For OP-ELM, the activation function used was Gaussian function. The network structures of ELM networks have been determined by trial and error method with grid search technique.

4.1 Comparison of EEM-ELM and EM-ELM with the same expected accuracy

We compare the network structure obtained by both EEM-ELM and EM-ELM with the same stopping RMSE for regression cases and stopping rate for classification cases. Table 2 shows the network structure and generalization performance of both EEM-ELM and EM-ELM network for all six applications. In this section, we only compare the performance between EM-ELM and EEM-ELM with $k = 10$ (denoted EEM-ELM(10)).

From the table, it shows that with comparable generalization performance, EEM-ELM(10) always has lower network complexity as compared to EM-ELM. While, it is obvious that both EM-ELM and EEM-ELM lead to smaller network than original ELM. In addition, the testing standard deviation (Std dev) of EEM-ELM(10) is better than or comparable with EM-ELM, which shows the stability of EEM-ELM. Fig. 1 shows the performance comparison of EEM-ELM(10) and EM-ELM with sigmoidal additive nodes on Abalone case. To achieve the same testing RMSE (0.08), EM-ELM needs 12 hidden nodes, while EEM-ELM(10) only needs 7 hidden nodes. Similar curves could be found for other applications.

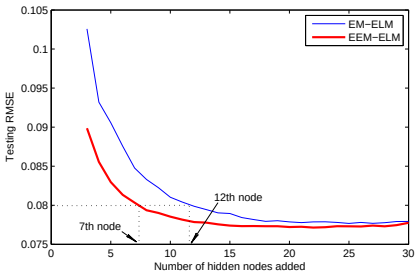

Fig. 1: Performance comparison of EEM-ELM(10) and EM-ELM with sigmoidal additive hidden nodes on Abalone case

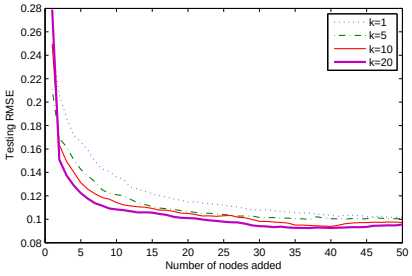

Fig. 2: Performance comparison of EEM-ELM with different values of k with sigmoidal additive hidden nodes on Boston Housing case

4.2 Effect of factor k

In this section, the effect of number of hidden nodes generated for selection k is studied. From Table 2, we compare the performance between EEM-ELM(10) and EEM-ELM with $k = 20$ (denoted EEM-ELM(20)). The table shows that EEM-ELM(20) achieves comparable generalization performance with even more compact network structure. And EEM-ELM(20) is stable according to the testing standard deviation. Fig. 2 shows the performance of EEM-ELM with different values of k with sigmoidal additive nodes on Boston Housing case. The conclusion above is further proved in Fig. 2.

5 Conclusion

This paper proposes an enhancement of EM-ELM (referred to as EEM-ELM), which introduces a selection phase based on the random search method. Unlike in original EM-ELM, at each incremental learning step, several hidden nodes are randomly generated and the hidden node that leads to highest residual error reduction will be added to the network. Empirical studies show that EEM-ELM with a much smaller network can achieve better or similar performance as that of EM-ELM for the six cases we studied, and EEM-ELM is faster and achieves more compact network as compared to OP-ELM for regression cases (OP-ELM toolbox does not support the multi-class classification cases. Hence, there is no result shown for two classification cases).

References

[1] G. R. Feng, G.-B. Huang, Q. P. Lin, and R. Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357, 2009.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.

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

Table 2: Performance comparison between EEM-ELM, EM-ELM ELM and OP-ELM (Trial&Error time is the total time spent in determining the network structure of ELM network. Training time is the average time spent on each round of simulation. #Nodes for OP-ELM is the maximum number of nodes for selection)

Datasets (Stop RMSE)	Algorithms(k)	#Nodes	Trial&Error Time (s)	Training Time (s)	Testing RMSE	Testing Std dev
Abalone (0.08)	ELM	35	411.0625	0.0164	0.0772	0.0017
	EM-ELM(1)	11.7		0.0148	0.0804	0.0028
	EEM-ELM(10)	7.1		0.0563	0.0802	0.0027
	EEM-ELM(20)	6.5		0.0992	0.0802	0.0023
	OP-ELM	100		3.5898	0.0783	0.0031
Auto-MPG (0.07)	ELM	25	38.2813	0.0016	0.0788	0.0047
	EM-ELM(1)	19.55		0.0109	0.0798	0.0055
	EEM-ELM(10)	14.65		0.057	0.0787	0.0056
	EEM-ELM(20)	13.1		0.0898	0.0784	0.0045
	OP-ELM	100		0.2625	0.0821	0.0080
Boston Housing (0.07)	ELM	50	77.2188	0.0047	0.1007	0.0115
	EM-ELM(1)	49.95		0.025	0.1058	0.016
	EEM-ELM(10)	30.7		0.1234	0.0978	0.0108
	EEM-ELM(20)	26.2		0.207	0.0973	0.0124
	OP-ELM	100		0.2969	0.0901	0.0086
Machine CPU (0.03)	ELM	10	18.7188	0.0008	0.0689	0.0189
	EM-ELM(1)	13.9		0.0031	0.0861	0.061
	EEM-ELM(10)	9.05		0.0313	0.0728	0.0301
	EEM-ELM(20)	8.55		0.057	0.0746	0.0438
	OP-ELM	50		0.0437	0.0764	0.0181
Datasets (Stop Rate)	Algorithms(k)	#Nodes	Trial&Error Time (s)	Training Time (s)	Testing Rate	Testing Std dev
Image Segmentation (90%)	ELM	210	815.6875	0.1953	94.83%	0.0057
	EM-ELM(1)	92.6		0.7477	93.03%	0.0108
	EEM-ELM(10)	69.6		4.6758	93.26%	0.0077
	EEM-ELM(20)	60.9		8	92.82%	0.0081
Satellite Image (90%)	ELM	520	4328.3	2.9648	89.26%	0.0036
	EM-ELM(1)	165		4.9172	87.64%	0.0065
	EEM-ELM(10)	126.6		30.4555	87.49%	0.0056
	EEM-ELM(20)	117.25		54.1523	87.23%	0.0046

[4] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71:3060–3068, 2008.

[5] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70:3056–3062, 2007.

[6] R. E. Cline. Representations for the generalized inverse of a partitioned matrix. *Journal of the Society for Industrial and Applied Mathematics*, 12(3):588–600, 1964.

[7] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[8] Y. Miche, A. Sorjamaa, and A. Lendasse. OP-ELM: Theory, experiments and a toolbox. In V. Kurková, R. Neruda, and J. Koutník, editors, *LNCS - Artificial Neural Networks - ICANN 2008 - Part I*, volume 5163/2008 of *Lecture Notes in Computer Science*, pages 145–154. Springer Berlin / Heidelberg, September 2008.

[9] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. OP-ELM: Optimally-pruned extreme learning machine. *IEEE Transactions on Neural Networks*, to appear.