Design of an ensemble neural network to improve the identification performance of a gas sweetening plant using the negative correlation learning and genetic algorithm

Article *in* Journal of Natural Gas Science and Engineering · November 2014

DOI: 10.1016/j.jngse.2014.07.012

CITATIONS
9

READS
109

3 authors:

Javad Sadeghi Azizkhani
University of Saskatchewan
2 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Hooshang Jazayeri-Rad
Petroleum University of Technology
37 PUBLICATIONS **393** CITATIONS
SEE PROFILE

Nader Nabhani
Petroleum University of Technology
37 PUBLICATIONS **281** CITATIONS
SEE PROFILE

Journal of Natural Gas Science and Engineering 21 (2014) 26–39

Contents lists available at ScienceDirect

Journal of Natural Gas Science and Engineering

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

Design of an ensemble neural network to improve the identification performance of a gas sweetening plant using the negative correlation learning and genetic algorithm

Javad Sadeghi Azizkhani [a], Hooshang Jazayeri-Rad [a, *], Nader Nabhani [b]

[a] Department of Automation and Instrumentation, Petroleum University of Technology, Ahwaz, Iran
[b] Department of Mechanical Engineering, Petroleum University of Technology, Ahwaz, Iran

ARTICLE INFO

Article history:
Received 8 May 2014
Received in revised form
10 July 2014
Accepted 12 July 2014
Available online

Keywords:
Ensemble neural network
Negative correlation learning
Genetic Algorithm
Gas sweetening
H_2S content
Alkanoamine

ABSTRACT

This paper presents a combination of negative correlation learning (NCL) and Genetic Algorithm (GA) to create an ensemble neural network (ENN). In this approach the component neural networks (CNNs) of ENN are trained simultaneously. The resulting CNNs negatively correlate together through the penalty terms in their objective functions. The predicted output is obtained by using the weighted averaging of the outputs of CNNs. GA participates in the training of CNNs and assigns proper weights to each trained CNN in the ensemble. The proposed method was tested on a case study in the Gas Treatment Plant (GTP) of the AMMAK project in the Ahwaz onshore field in Iran. The testing results of the model properly follow the experimental data. In addition, the proposed method outperformed the single neural network and some other network ensemble techniques.

1. Introduction

Today system identification is one of the most important fields of research in engineering sciences. The identification is a process of deriving a model of a physical system by using a series of experimental data. One of the most recent and authoritative techniques for identification problems is Artificial Neural Network (ANN). Main reasons of popularity of this approach are simplifying the process modeling and enabling implementation of inclusive tools for control system design. However, most chemical processes are complicated and a Single Neural Network (SNN) cannot identify such a complex system accurately. To overcome this limitation, improve the performance of ANN and reduce the model output variance, Hansen and Salamon (1990) proposed ENN which is a combination of a series of CNNs. The key point in applying these individual networks is their dissimilarity and diversity. In other word, there is no advantage in aggregation of networks which are all identical (Cunningham et al., 2000). There are many ensemble techniques developed by researchers, most of these techniques follow two steps: first generating individual networks, and then combining them (Sharkey, 1996). In such techniques, CNNs are learned independently. Two disadvantages of these approaches are loss of interaction between independent CNNs during training (Liu and Yao, 1999) and possibility of the lack of proper distribution of independent networks throughout the ensemble. Two of the most popular methods of this approach of ENN construction are Bagging (*B*ootstrap *Agg*regating) proposed by Breiman (1996) and Boosting proposed by Freund and Schapire (1996). Another recent pioneer technique in ENN employs the NCL method. NCL is different from previous works which trained the individual networks independently or sequentially (Drucker et al., 1994). Rather than producing unbiased individual networks whose errors are uncorrelated, NCL can create negatively correlated networks to encourage specialization and cooperation among the individual networks (Liu and Yao, 1999). By correlating individual networks through the NCL method, assigning a proper weight to each CNN by GA and combining the CNN outputs through weighted averaging, we create an ENN which accurately characterizes the process. An empirical study was performed in this paper on a regression problem of a

* Corresponding author. Automation and Instrumentation Department, Petroleum University of Technology, 63431 Ahwaz, Iran. Tel.: +98 919 611 3440.
E-mail address: Jazayeri.sadeghi611@gmail.com (H. Jazayeri-Rad).

http://dx.doi.org/10.1016/j.jngse.2014.07.012
1875-5100/

chemical process to show the ability of the negative correlation learning and genetic algorithm to identify and model a complicated system. The rest of this paper is organized as follows: Section 2 describes the negative correlation learning and genetic algorithm. As an empirical case study, Section 3 describes an experimental study of the negative correlation learning of a nonlinear chemical process in a refinery in AHWAZ petroleum field. Section 4 specifies the input and output variables of CNNs. Section 5 of the paper presents results of this study. Summary and conclusion are provided in Section 6.

2. Combination of the negative correlation and Genetic Algorithm

To learn an ensemble network, there should be a plan to create dissimilar and diverse networks and another plan to combine the outcomes of these diverse networks in order to strengthen accurate networks and weaken poor ones in the ENN output. Diversity of networks is important because there is no advantage in multiplying similar networks which generalize identically. Different training parameters (Hansen and Salamon, 1990), different training patterns (Bauer and Kohavi, 1999), different feature subsets (Zio et al., 2008) and different learning methods for each network of the ensemble (Xu et al., 1992) are among some techniques of creating diverse networks. In addition, simple averaging and weighted averaging are famous methods of combining CNNs. In this paper, we use the different feature subset technique to generate diversity and the weight averaging method for combining CNNs.

Now suppose that the experimental input matrix, X, and output matrix, $F(X)$, are defined as:

$$X = [X_1, X_2, X_3, X_4, X_5]_{Z\times 5} \tag{1}$$

$$F(X) = [F(X_1, X_2, X_3, X_4, X_5)]_{Z\times 1} \tag{2}$$

where $X_Z \in R$ and Z is the size of the data set. This section considers the estimation of $F(x)$ by creating a negative correlation ensemble neural network (NCENN) which increases accuracy and reduces the generalization error. Firstly, we train 'M' individual neural networks and then combine them. These individual networks are dissimilar in their training data sets. This results in different base learners for the individual CNN and increases their diversity. In order to construct these sub-data sets the bootstrap method introduced by Efron (1979) is used. This is a general resampling method for estimating the distributions of data based on independent observations.

The proposed technique in this work simultaneously provides diverse and negative correlations amongst individual component networks. The predicted results are then compared with the results obtained using other methods of designing ENNs. In negative correlation learning, all the individual networks in the ensemble are trained simultaneously through the correlation penalty terms in their error functions (Liu and Yao, 1999). Therefore, by increasing the interactions between the component networks, the method trains each network to obtain the best outcome for the whole of ensemble.

We use the Levenberg–Marquardt Algorithm (LMA) to learn weights of CNNs by assigning negative correlation penalty terms in their objective functions. The LMA is a very precise and popular curve-fitting algorithm used in neural networks for solving generic curve-fitting problems. However, like many fitting algorithms, LMA finds only a local minimum which is not necessarily the global minimum. Our goal is to derive the global minimum of a negative correlation objective function. We employ GA to estimate the primal weights which directs the objective function to its global minimum point and then use LMA to derive accurate weights corresponding to this global minimum.

As shown in Fig. 1, LMA cannot compute weights which globally minimize the objective function, because it cannot distinguish between local and global minima. In such a situation, GA avoids these local minima by estimating the primal weights of each diverse component network. To achieve this goal, each network use 80% of the bootstrap sample sub-data set for training and the remaining 20% for testing. These data are randomly distributed all over the sample sub-data sets. The following equation presents the error function used in this step:

$$E_{iGA}(x) = \frac{1}{Z}\sum (f_{iGA}(x) - F(x))^2 \tag{3}$$

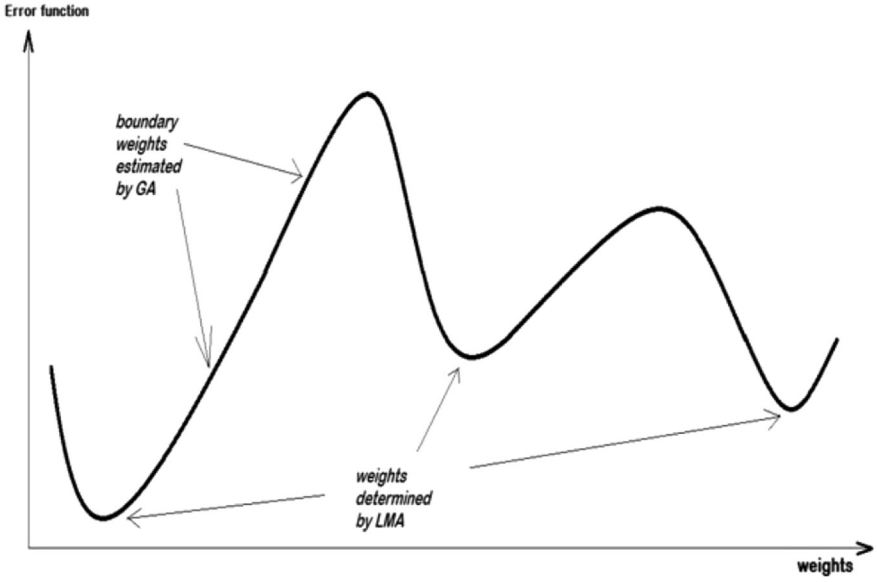

Fig. 1. Boundary of the weights estimated by GA and LMA.

where $E_{iGA}(x)$ is the objective function used to learn the primal weights by GA and $f_{iGA}(x)$ is the output of the ith network. The weights computed by GA generate outputs close to the global minimum. Consequently, we can use LMA to precisely tune the weights (W_i) using the following negative correlation objective function:

$$E_{iLMA}(x) = \frac{1}{Z}\sum_{i=1}^{M}(f_{iLMA}(x) - F(x))^2 - \frac{1}{Z}\lambda\sum_{i=1}^{M}P_i(x) \tag{4}$$

where $f_{iLMA}(x)$ is the output of the ith network and $E_{iLMA}(x)$ is the error function of the ith component neural network. The first term of this equation is the empirical error function and the second term is the negative correlation objective function. The term $P_i(x)$ is the correlation penalty function which negatively correlates output of each network to the average output of all networks. The parameter λ adjusts the strength of the penalty term. It is clear that when $\lambda = 0$, the networks are trained independently and there are not any negative correlations amongst them. This situation is similar to the conventional methods of designing ensemble neural network since there are not any interactions between networks.

The penalty function $P_i(x)$ has the form:

$$P_i(x) = \left(f_{iLMA}(x) - \overline{f(x)}\right)^2 \tag{5}$$

where $\overline{f(x)}$ is the average of all component network outputs obtained by:

$$\overline{f(x)} = \frac{\sum_{i=1}^{M} f_{iLMA}(x)}{M} \tag{6}$$

The value of λ should be within the range:

$$0 \le \lambda \le 1 \tag{7}$$

Precise weights and outputs of the diverse networks are generated after the network learning phase of CNNs using the combination of GA and LMA. A weight, W_i, is then assigned to each of the trained CNN outputs. Using GA, these network weights are further adjusted to minimize the ensemble generalization error function, $E_{\text{Ens.}}(x)$, defined as:

$$E_{\text{Ens.}}(x) = \frac{1}{Z}\sum_{i=1}^{M}\left[\sum_{i=1}^{M}(f_{iLMA}(x)*W_i') - F(x)\right]^2 \tag{8}$$

where W_i' must be within the range given in Eq. (9) and must satisfy the condition given in Eq. (10):

$$0 < W_i' < 1 \tag{9}$$

$$\sum_{i}^{N} W_i' = 1 \tag{10}$$

Using the weighted averaging, the final prediction of the output matrix of NCENN is presented as:

$$f_{\text{pred}}(x) = \sum_{i}^{M}(f_{iLMA}(x)*W_i') \tag{11}$$

Now we define the $Z \times M$ matrix $R_{Z\times M}$ each column of which represents the differences between the actual output data and the output data of the ith network:

$$R = F(x) - f_{iLMA}(x) \tag{12}$$

The correlation coefficient parameter between the kth and jth columns (or kth and jth networks) of matrix R is given by:

$$\text{Corr}_{kj} = \frac{C_{kj}}{\sqrt{C_{kk}\cdot C_{jj}}} \tag{13}$$

where C_{kj} is the covariance of the kth and jth networks and C_{kk} and C_{jj} represent the variances for the kth and jth networks, respectively. In addition, we have:

$$\text{Corr}_{kj} = \text{Corr}_{jk} \tag{14}$$

The NCENN algorithm developed in this work can be summarized as follows:

Given: $\{(X_1, F(X_1)), \ldots, (X_Z, F(X_Z))\}$ where X is the experimental input matrix and $F(X)$ is the output matrix

(i) Let M be the ultimate number of predictors required (obtained by trial and error)

(ii) For $m = 1$ to M

a) Make a training set TM by resampling N items at random with replacement from the data set (bootstrap method)

b) Train a network with the set TM and add it to the ensemble by using GA to compute the primal weights using the objective function:

$$E_{iGA}(x) = \frac{1}{Z}\sum(f_{iGA}(x) - F(x))^2 \tag{15}$$

c) Continue training the network with the set TM and add it to the ensemble by using LMA to compute the precise weights using the objective function:

$$E_{iLMA}(x) = \frac{1}{Z}\sum_{i=1}^{M}(f_{iLMA}(x) - F(x))^2 - \frac{1}{Z}\lambda\sum_{i=1}^{M}P_i(x) \tag{16}$$

(iii) Using GA to assign proper a weight to each of the trained CNNs outputs using the objective function:

$$E_{\text{Ens.}}(x) = \frac{1}{Z}\sum_{i=1}^{M}\left[\sum_{i=1}^{M}(f_{iLMA}(x)*W_i') - F(x)\right]^2 \tag{17}$$

(iv) For any new testing pattern, the NCENN ensemble output is computed using the equation:

$$f_{\text{pred}}(x) = \sum_{i}^{M}(f_{iLMA}(x)*W_i') \tag{18}$$

3. Experimental case study

Natural gas which is trapped in porous rock underground may have some Hydrogen Sulfide (H_2S) and Carbon Dioxide (CO_2) and is considered as sour gas. H_2S has drastic toxic properties and in the presence of water (H_2O) can form Sulfuric Acid (H_2SO_4) which is highly corrosive. So, it is necessary to remove the H_2S compound from the natural sour gas. H_2S removal which is called Gas Sweetening is necessary to avoid corrosion of combustion engines and SO_x generation in the flue gases (Fortuny et al., 2008). There are some treatment processes for removal of H_2S from natural gas in the gas sweetening plants. These processes include chemical solvents processes, physical solvents processes,

Table 1
Methods of treatment processes.

Removal mechanism	Process type	Technology	Commercial name
Chemical absorption	Regenerative, continuous	Amines	MEA, DEA, MDEA DIPA, DGA. formulated solvents
		Potassium carbonate	Benfield, Catacarb, Giammarco-vetrocooke, etc.
	Non Regenerative, continuous (usual arrangement lead/lag)	Sodium hydroxide	–
Physical absorption	Regenerative, continuous	Physical solvents	Selexol, Rectisol, Purisol, Fluor solvent, IFPexol, etc.
Physical–chemical absorption	Regenerative continuous	Physical-chemical solvents	Sullinol, Ucarsol LE 701, 702 and 703. Flexorb PS, etc.
Physical adsorption	Regenerative. continuous (adsorption/desorption sequence)	Molecular sieves	Z5A (Zeochem), LNG-3 (UOP), etc.
Permeation	Continuous	Membranes	Separex, Cynara, Z-lop, Medal, etc.

adsorption processes, hybrid solvents processes and physical separation processes (membrane) (Seqatoleslami et al., 2011). Each of these processes is briefly described in Table 1 (Maddox and Sheerar, 1994).

Today chemical solvents plants are more common techniques in petroleum industry because of their high efficiency in the removal of H_2S and relative easy and inexpensive regeneration of their solvents. In the past few years, mixed Alkanoamine solvents have received increased attention. MonoEthanolAmine (MEA), Di-EthanolAmine (DEA) and MethylDi-EthanolAmine (MDEA) are more common Alkanoamines used in the industry. DEA with the chemical formula of $(C_2H_5O)_2NH$ has a number of characteristics which make it more suitable than the other Alkanoamines. These characteristics are:

(i) The mole/mole loadings typically used with DEA (0.35–0.8 mole/mole) are much higher than those normally used for MEA and other Alkanoamines (0.3–0.4),
(ii) Because DEA does not form a significant amount of non re-generable degradation products, a re-claimer is not required,
(iii) DEA is a secondary Amine and is chemically weaker than MEA, and less heat is required to strip the Amine solution,
(iv) DEA forms a re-generable compound with H_2S and CO_2 and can be used for the partial removal of H_2S and CO_2 without significant solution losses (Abedinzadegan, 2008).

The chemical reaction in H_2S removal process is as follows:

$$2RNH_2 + H_2S \rightarrow (RNH_3)_2S$$

where R is the mono-, di- or tri-ethanol and N, H and S denotes nitrogen, hydrogen and sulfur, respectively. Chemical solvent gas treatment plants separate H_2S and CO_2 from other components of sour gas by Alkanoamines in the Amine contactor tower. For regeneration of this rich Alkanoamines, it is flowed to Amine regenerator tower and after heating it, lean Alkanoamines are recycled to the Amine contactor tower again. Contactor towers work in high pressure and low temperature and regenerator towers work at low pressure and high temperature. The Amine regenerator reboilers supply the necessary heat to strip H_2S and CO_2 from the rich Amine using steam as the heating medium (Yao et al., 2005). Regenerated liquid-phase lean Amine exiting from the bottom of tower is then pumped to the Amine contactor and the vapor-phase Amine exiting from the top of the tower is then cooled, condensed and refluxed back to regenerator tower. The recovered hydrogen sulfide gas stream (acid gas) may then be: (i) vented; (ii) flared in waste gas flares or modern smokeless flares; (iii) incinerated; or d) utilized for the production of elemental sulfur or sulfuric acid. Fig. 2 shows the schematic diagram of the Amine regenerator tower.

A suitable model of the Alkanoamine regeneration tower using the NCENN technique can be used to enhance the plant analysis and design through economically providing appropriate responses under diverse plant conditions. Models make it possible to predict a system's behavior. An accurate model in a plant optimization program can increase the production yield and decrease the environmental pollution. Modeling may also be performed for other purposes such as: monitoring of processes, designing controllers, fault detection and component diagnosis (Isermann and Munchhof, 2009).

4. Specifying the input and output variables and training of CNNs

Lean Amine exited from the bottom of the regeneration tower is cycled back to the absorption tower. If the H_2S content of the lean Amine is lowered, the removal of H_2S and CO_2 from the sour gas is increased. So production of lean Amine with the minimum H_2S content is one of the objectives of gas sweetening plants. The value of this parameter at time (t) is given by the output of the dynamic model.

The operation performance of Alkanoamine regeneration tower is largely affected by the pressure and temperature parameters. Maximum stripping of H_2S and CO_2 content of rich Amine occurs in low pressure and high temperature conditions. The steam load entered to the re-boilers lowers the partial pressures of H_2S and CO_2 content in the gas stream and increases

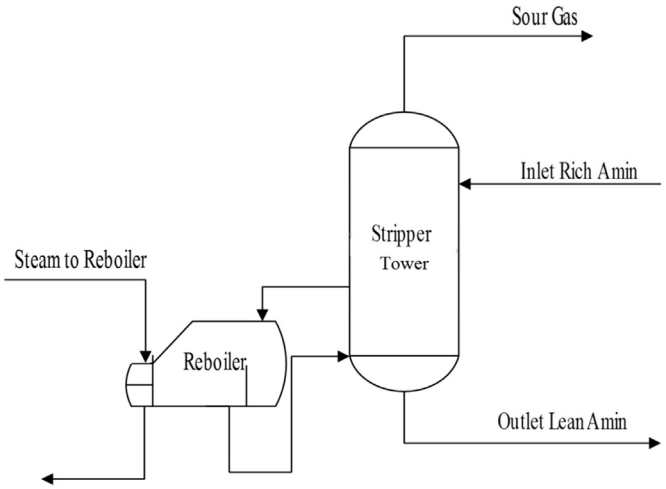

Fig. 2. Schematic diagram of the Amine regenerator tower.

the separating forces of these acid gases from the Amine solution. Another important parameter which influences the efficiency of the tower is the temperature of the inlet rich Amine. An increase of the temperature of the rich Amine causes more acid gas stripping to be occurred in the overhead trays. Therefore, this results in less steam loading and less energy consumption. To generate a dynamic model, the samples of the steam load entered to the re-boilers and the temperatures of the inlet rich Amine at times (t) and $(t-1)$ are employed as the inputs of the model. In addition, the sample of H_2S content of lean Amine outlet from the bottom of the regeneration tower at time $(t-1)$ is fed to the input of the model.

The network parameters in this paper are summarized as follows: one hidden layer with 12 neurons, log-sigmoid and linear functions as the activation functions in the hidden and output layers, respectively, 500 epochs for the training procedure, identical initial weights to be used for the learning cycle, 80% of the sub-data sets (selected randomly) for training and 20% for testing, the Levenberg–Marquardt learning algorithm with negative correlation objective function, GA as a learning method which assigns proper weights to each network and finally the weighted averaging method is used to combine results and create ENN. These networks become diverse by employing dissimilar sub-data sets generated by the bootstrap resampling method. It should be noted that the selection of 80% of the database for training will probably provide enough samples representing the dynamics of the system in the required dynamic range. The benefit of this kind of data allocation is that in each subset there is enough representative data for the whole ranges of operating conditions (Fayazi et al., 2014). It is worth noting that the soft computing models should be tested inside the employed data domain (Haykin, 1994). The developed models perform well within the trained domains (interpolation). However, the accuracy will be reduced if these developed models are tested using the data outside the operating range (extrapolation).

Ensemble neural network using 10 correlated and diverse component neural networks is used to identify the regenerator tower of GTP of the "AMMAK" project in Ahwaz Iranian onshore field. The model is dynamic and past inputs and outputs affect the current output. The total data employed in this work consisted of 1600 data points which were collected during six months of the plant operation. These data represent all possible plant conditions. The resulting model prediction is highly nonlinear.

5. Results

To evaluate the performances of the algorithms developed in this paper and compare them with other methods or architectures, mean square error (MSE), mean absolute error (MAE) and standard deviation (STD) were calculated by using the following equations:

$$\mathrm{MSE} = \frac{\sum_{j=1}^{Z}\left(F_j(x) - f_{\mathrm{pred},j}(x)\right)^2}{Z} \tag{19}$$

$$\mathrm{MAE} = \frac{\sum_{j=1}^{Z}\left|F_j(x) - f_{\mathrm{pred},j}(x)\right|}{Z \times \beta^2} \tag{20}$$

$$\mathrm{STD} = \sqrt{\frac{\sum_{j=1}^{Z}\left(f_{\mathrm{pred},j}(x) - \overline{f_{\mathrm{pred}}(x)}\right)^2}{Z}} \tag{21}$$

where Z is the size of data set, $F_j(x)$ is the jth experimental data, $f_{\mathrm{pred},j}(x)$ is the jth predicted data, β^2 is the variance of the experimental data and $\overline{f_{\mathrm{pred}}(x)}$ is the simple averaging of predicted data, i.e.

$$\overline{f_{\mathrm{pred}}(x)} = \frac{\sum_{j=1}^{Z} f_{\mathrm{pred},j}(x)}{Z} \tag{22}$$

The MSE is equal to the sum of the variance and the squared bias of the estimator:

$$\mathrm{MSE} = \mathrm{variance} + \mathrm{bias}^2 \tag{23}$$

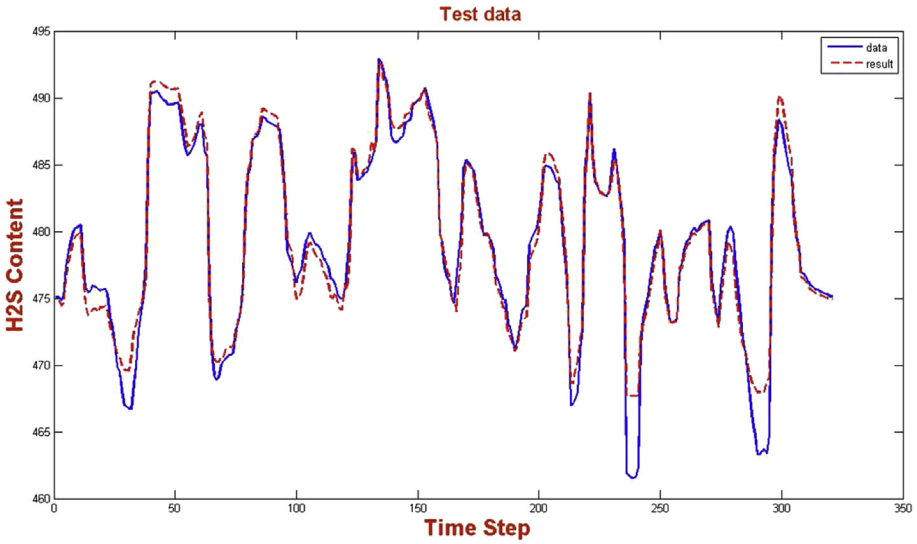

Fig. 3. Exclusively using GA to learn networks.

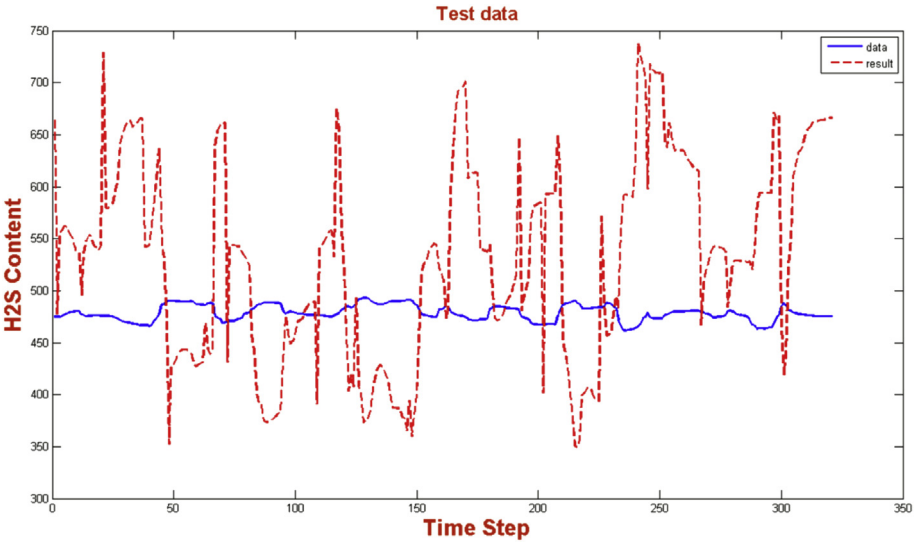

Fig. 4. Exclusively using LMA to learn networks (number of program runs = 18).

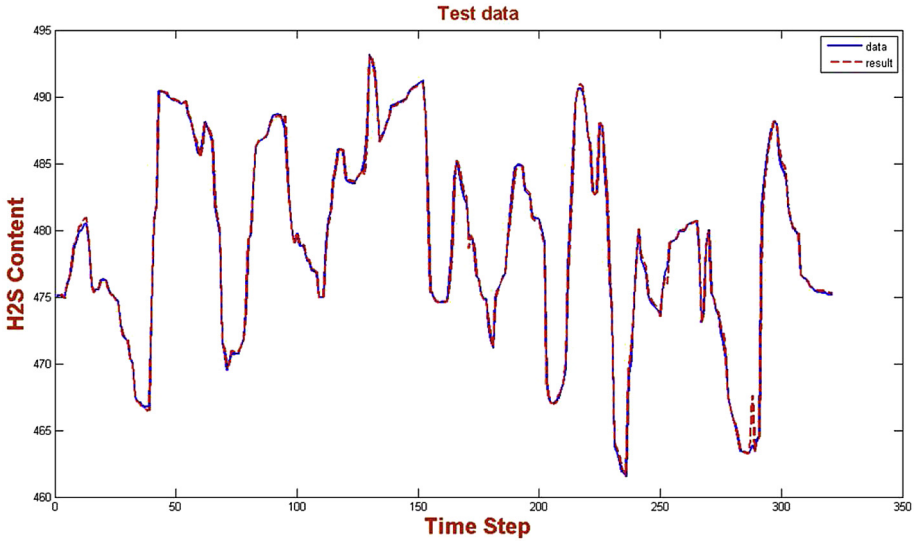

Fig. 5. Exclusively using LMA to learn networks (number of program runs = 24).

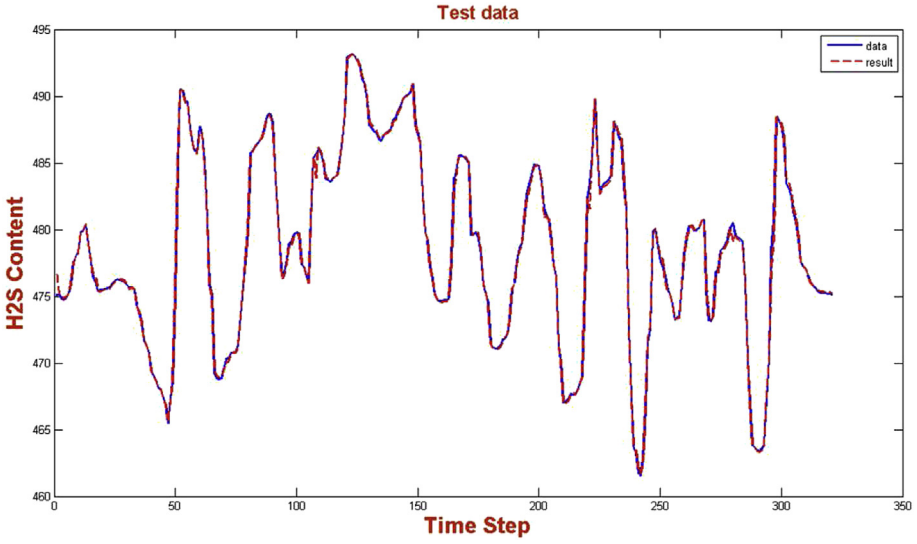

Fig. 6. Using the combination of GA and LMA to learn networks.

Table 2
Performances of ENN using different learning methods of networks.

	GA	LMA	GA + LMA
MSE	0.0388	0.0021	0.0015
MAE	0.0427	7.2144e−004	5.3963e−004
STD	0.0036	4.983e−004	2.2686e−004

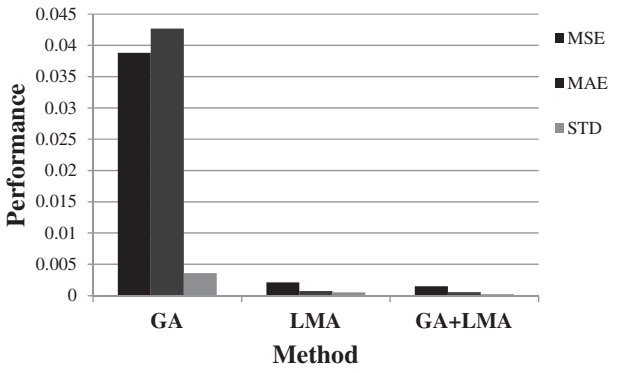

Fig. 7. Schematically comparing the performances of ENN with different learning methods of networks.

So by minimizing MSE, bias and variance are generally decreased. MAE is another network performance function. It measures the network performance as the mean of absolute errors.

As mentioned in Section 2, in the first step of learning each network, we use GA to estimate the primal weights. This action prevents the algorithm from being trapped into local minima of the objective function. Consequently, the application of LMA improves the accuracy of the global minimum. Figs. 3–6 present the results and Table 2 compares the performances of the developed ENN under the following situations:

(i) Exclusively using GA as the learning algorithm for training each network;
(ii) Exclusively using LMA as the learning algorithm for training each network;
(iii) Using the combination of GA and LMA as the learning algorithm for training each network.

As we see, GA can't accurately identify the system, because it is not precise enough. It just leads us to the correct path. LMA with $\lambda = 0.5$ is used for situations (i) and (ii) above. LMA can't distinguish local and global minima. Therefore, by utilizing a large computational load, we repeated the execution of our

Table 3
Values of the correlation coefficient parameter for $\lambda = 0.0$.

$\lambda = 0$	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9	NN10
NN1	1	0.790172	0.792748	0.748006	0.866052	0.806097	0.917694	0.806057	0.798721	0.811676
NN2	0.790172	1	0.856833	0.851513	0.815966	0.810149	0.785917	0.809992	0.866033	0.814132
NN3	0.792748	0.856833	1	0.954977	0.784262	0.880325	0.800138	0.880512	0.847137	0.843714
NN4	0.748006	0.851513	0.954977	1	0.744552	0.849878	0.757816	0.850101	0.813546	0.830532
NN5	0.866052	0.815966	0.784262	0.744552	1	0.880224	0.805651	0.880143	0.878347	0.72579
NN6	0.806097	0.810149	0.880325	0.849878	0.880224	1	0.79398	0.999998	0.9002	0.742236
NN7	0.917694	0.785917	0.800138	0.757816	0.805651	0.79398	1	0.793869	0.830857	0.862573
NN8	0.806057	0.809992	0.880512	0.850101	0.880143	0.999998	0.793869	1	0.900017	0.742265
NN9	0.798721	0.866033	0.847137	0.813546	0.878347	0.9002	0.830857	0.900017	1	0.792304
NN10	0.811676	0.814132	0.843714	0.830532	0.72579	0.742236	0.862573	0.742265	0.792304	1

Table 4
Values of the correlation coefficient parameter for $\lambda = 0.6$.

$\lambda = 0.6$	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9	NN10
NN1	1	−0.8658	−0.37147	0.810166	0.930566	0.056484	−0.22077	0.880366	−0.56074	−0.52978
NN2	−0.8658	1	0.426638	−0.66412	−0.87463	0.172366	−0.06638	−0.83972	0.363203	0.290387
NN3	−0.37147	0.426638	1	−0.32109	−0.30336	0.551704	−0.26272	−0.45647	−0.18465	−0.1633
NN4	0.810166	−0.66412	−0.32109	1	0.831754	0.097146	−0.42464	0.819862	−0.60688	−0.58446
NN5	0.930566	−0.87463	−0.30336	0.831754	1	−0.00212	−0.28291	0.896456	−0.5461	−0.53273
NN6	0.056484	0.172366	0.551704	0.097146	−0.00212	1	−0.36015	−0.01728	−0.51537	−0.68551
NN7	−0.22077	−0.06638	−0.26272	−0.42464	−0.28291	−0.36015	1	−0.2598	0.374437	0.594388
NN8	0.880366	−0.83972	−0.45647	0.819862	0.896456	−0.01728	−0.2598	1	−0.4819	−0.59144
NN9	−0.56074	0.363203	−0.18465	−0.60688	−0.5461	−0.51537	0.374437	−0.4819	1	0.622782
NN10	−0.52978	0.290387	−0.1633	−0.58446	−0.53273	−0.68551	0.594388	−0.59144	0.622782	1

Table 5
Values of the correlation coefficient parameter for $\lambda = 1.0$.

$\lambda = 1.0$	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9	NN10
NN1	1	−0.73211	−0.11094	−0.77462	−0.80902	−0.68424	−0.27177	−0.02112	−0.28569	−0.10777
NN2	−0.73211	1	−0.32174	−0.90052	−0.75052	−0.77265	−0.36314	−0.20669	−0.23114	−0.32155
NN3	−0.11094	−0.32174	1	−0.09258	−0.27387	−0.06877	−0.01735	−0.21311	−0.10935	−0.23923
NN4	−0.77462	−0.90052	−0.09258	1	−0.75366	−0.85199	−0.39147	−0.26901	−0.23817	−0.23348
NN5	−0.80902	−0.75052	−0.27387	−0.75366	1	−0.55664	−0.35056	−0.19574	−0.42758	−0.00738
NN6	−0.68424	−0.77265	−0.06877	−0.85199	−0.55664	1	−0.15599	−0.56482	−0.00812	−0.32525
NN7	−0.27177	−0.36341	−0.01735	−0.39147	−0.35056	−0.15599	1	−0.10724	−0.54397	−0.09007
NN8	−0.02112	−0.20669	−0.21311	−0.26901	−0.19574	−0.56482	−0.10724	1	−0.32490	−0.54177
NN9	−0.28569	−0.23114	−0.10935	−0.23817	−0.42758	−0.00812	−0.54397	−0.32490	1	−0.28562
NN10	−0.10777	−0.32155	−0.23923	−0.23348	−0.00738	−0.32525	−0.09007	−0.54177	−0.28562	1

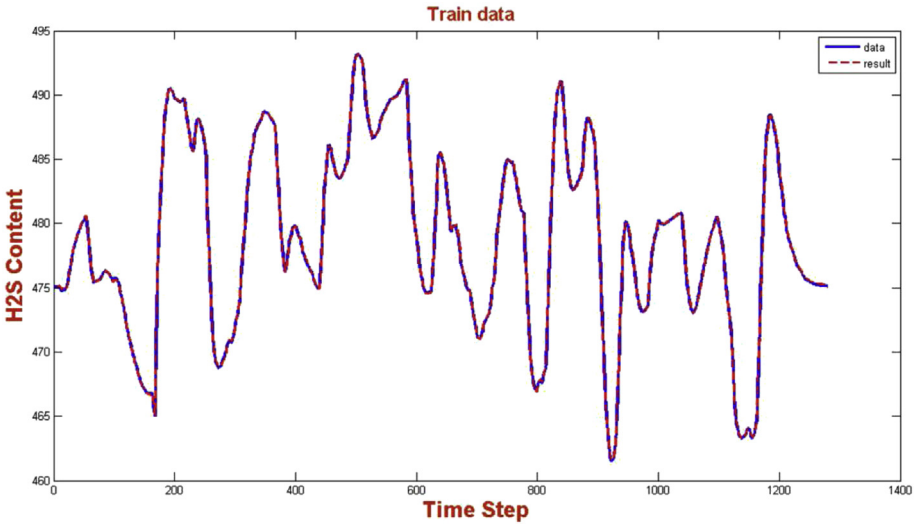

Fig. 8. Training data for $\lambda = 0$.

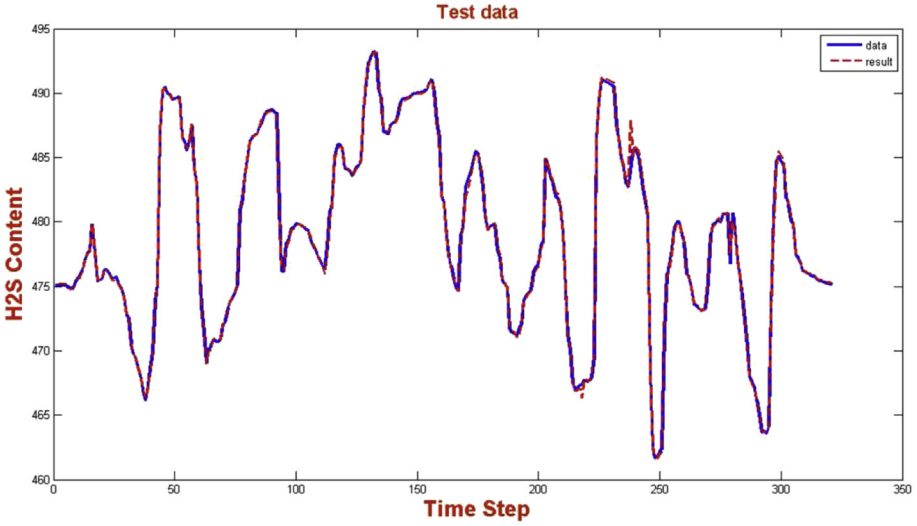

Fig. 9. Testing data for $\lambda = 0$.

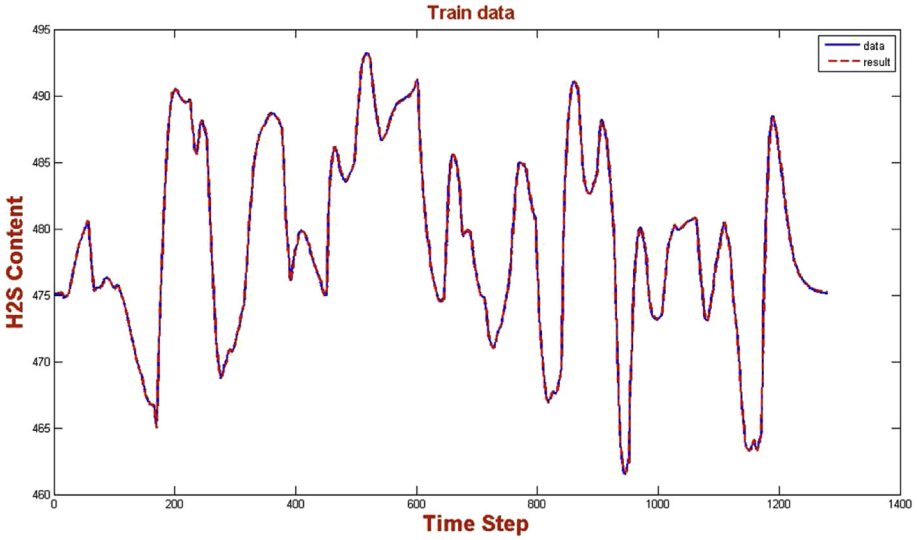

Fig. 10. Training data for $\lambda = 0.6$.

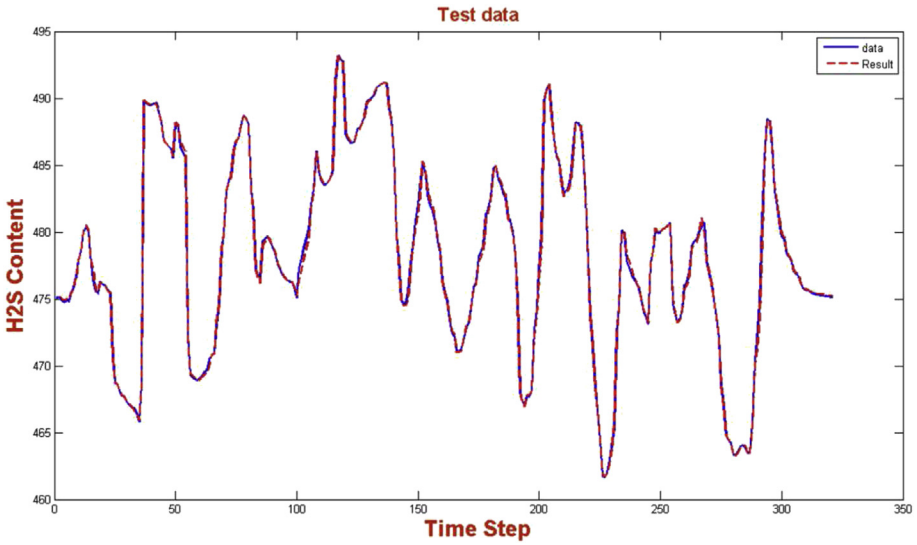

Fig. 11. Testing data for $\lambda = 0.6$.

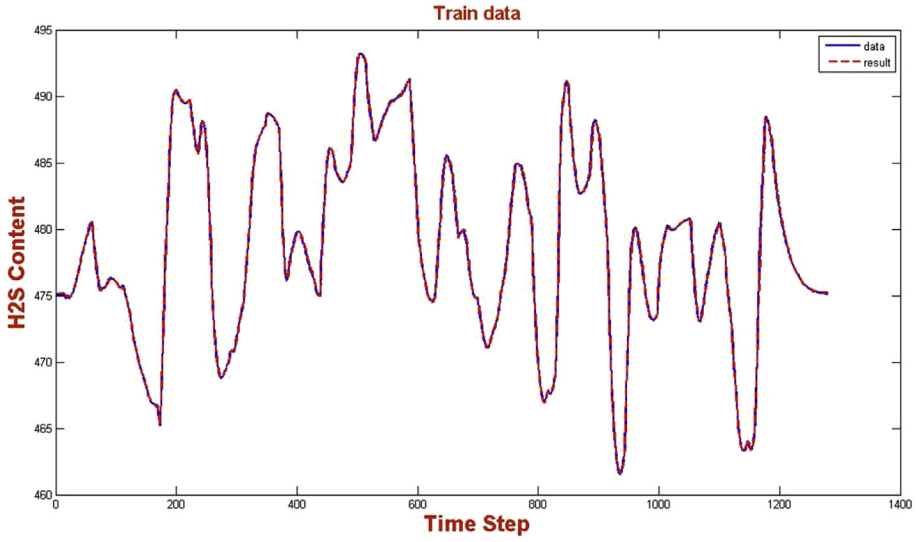

Fig. 12. Training data for $\lambda = 1$.

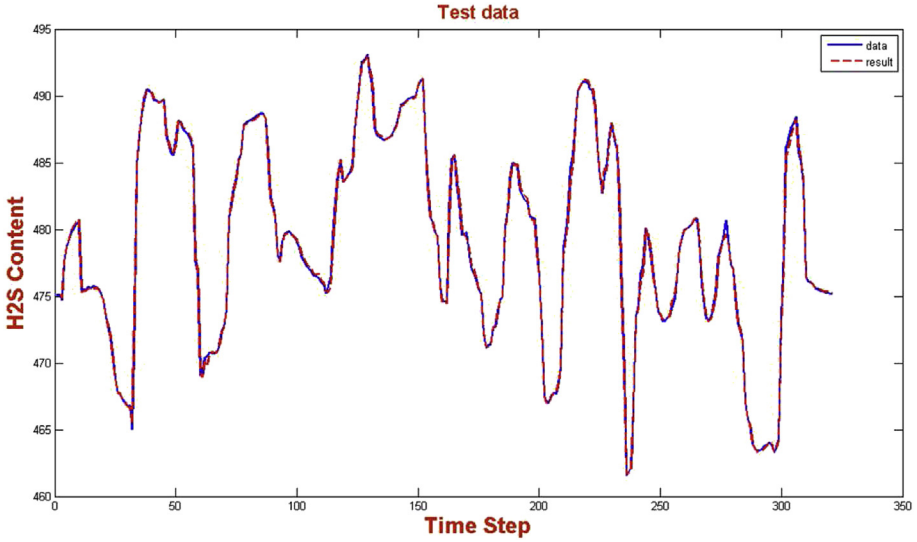

Fig. 13. Testing data for $\lambda = 1$.

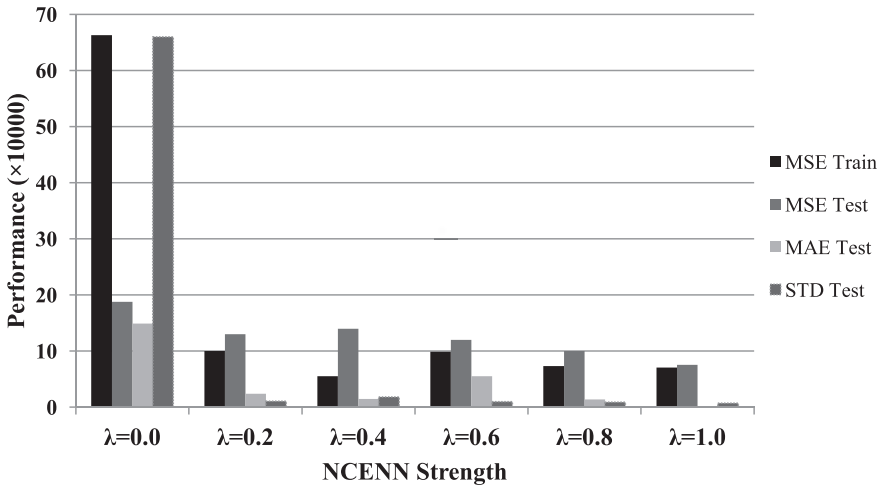

Fig. 14. Schematically comparing the performances of NCENN with different values of λ.

Table 6
Performances of NCENN with different values of λ.

	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
MSE train	0.00663	0.00100	5.52E−04	7.31E−04	9.89E−04	7.05E−04
MSE test	0.00188	0.00170	0.00140	0.00120	0.00100	7.53E−04
MAE test	0.00149	2.4E−04	1.49E−04	5.54E−04	1.40E−04	1.34E−05
STD test	0.00660	1.08E−04	1.82E−04	9.78E−05	9.02E−05	7.26E−05

Table 7
Performance values of ensemble methods and SNN.

	NCENN with $\lambda = 1.0$	NCENN with $\lambda = 0.5$	Bagging	SNN
MSE train	7.05E−04	8.14E−04	0.00106	0.01152
MSE test	7.53E−04	0.00138	9.38E−04	0.00153
MAE test	1.34E−05	1.64E−04	3.7242E−04	7.39E−04
STD test	7.26E−05	1.97E−04	9.0523E−05	8.24E−04

program 24 times to obtain Fig. 5 and the corresponding results in Table 2. By combining GA and LMA, the computing load is decreased and the performance is increased simultaneously as shown in Table 2 and Fig. 7. Hence, the proper identification of experimental model became possible.

The correlation coefficient parameter presents measure of dependency of networks to each other. In Tables 3–5 the correlation coefficient parameters between networks, as given by Eq. (13), are exhibited. When $\lambda = 0$, $Corr_{kj}$ has relatively large positive values and networks are trained independently. However, by increasing λ, networks tend to correlate more negatively to each other. Finally for $\lambda = 1$ the values of all correlation coefficient parameters become negative values which means that all networks are negatively correlated to each other. These indicate the strength of the penalty function in Eq. (4).

Figs. 8–13 show the corresponding identification results for $\lambda = 0.0$, 0.6 and 1.0, respectively. By comparing these results, it would become obvious that whenever networks correlate more negatively errors decrease, performance increase and the predicted

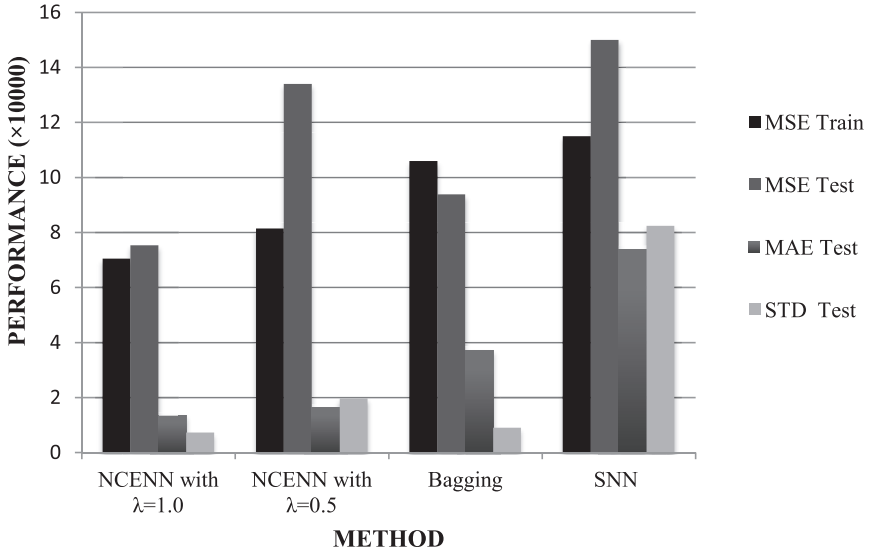

Fig. 15. Schematically comparing the performances of ensemble methods and SNN.

Table 8
Computational loads of ensemble methods and SNN.

	NCENN with $\lambda = 1.0$	NCENN with $\lambda = 0.5$	Bagging	SNN
Computing load time (s)	116.37	75.10	44.34	11.32

results follow experimental data more accurately. These observations are summarized in Fig. 14 and Table 6.

So far, the effect of negative correlation of individual networks on the performance of the whole ensemble and its ability to reduce errors is discussed. Now we learn our data using the bagging method and an SNN. These networks are then compared to the developed NCENN by taking into account their computational load and other network performances.

Bagging is one of the earliest methods of ensembling introduced by Breiman (1996). The idea behind bagging is to create independent networks from the same data set (Breiman, 1996). Bagging reduces variance or model inconsistency over diverse data sets from a given distribution, without increasing bias, which results in a reduced overall generalization error and enhanced stability. The other benefit of using bagging is related to the model selection which is no longer required (Shafiei and Jazayeri-Rad, 2012). The component networks should employ different data sets. To create dissimilar data sets, the same procedure as used in the first step of NCENN was employed. The procedure constructs 10 sub-data sets by the utilization of the bootstrap resampling method.

In our bagging method, 80% of sub-data set are used for training and the remaining 20% of the sub-data is used for testing. Ten independent networks are used and the weights of each network are initiated randomly. One hidden layer with 12 neurons and an activation function of tangent-sigmoid and linear transfer function as the activation function of the output layer are considered for each network. The networks are trained using the Levenberg–Marquardt back-propagation learning algorithm. Finally, simple averaging is used to combine the results of all networks. To construct SNN we used 80% of data set for training and 20% for testing, one hidden layer with 12 neurons and an activation function of log-sigmoid, the Levenberg–Marquardt as the learning algorithm and linear transfer function as the activation function of the output layer.

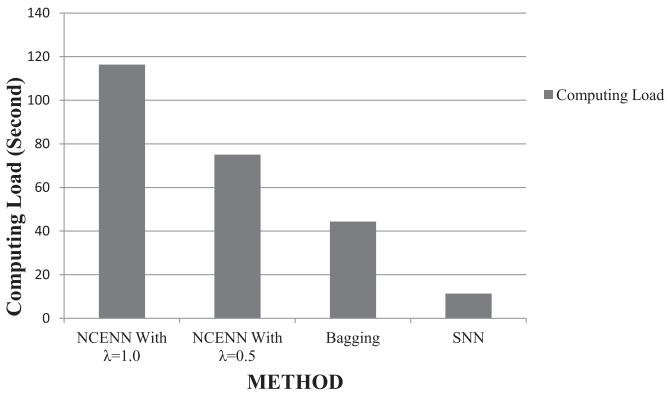

Fig. 16. Schematically comparing the computational loads of ensemble methods and SNN.

Table 7 presents the performances of NCENN, the bagging method and SNN. Fig. 15 compares the performances of these three networks schematically. By analyzing these results, it can be observed that errors are remarkably reduced by ensembling networks. NCENN with different values of the λ parameter remarkably outperforms the bagging method. It can be seen that by increasing the level of negative correlation, the interactions amongst networks will increase and consequently will result in the improvement of the identification performance.

Computational load time in engineering calculations is an important factor. Our aim is to identify the system within the shortest possible time. Table 8 gives the computational loads of different approaches employed in this work. Fig. 16 schematically compares the computational loads of the methods discussed in this paper. By analyzing these results, as we expected, SNN because of its simple architecture requires the minimum computing load. The bagging method because of the lack of interactions between their networks during learning requires a computing load less than NCENN. Finally, NCENN with $\lambda = 1.0$ necessitates a computing load more than the case where $\lambda = 0.5$ because of the highest correlation interactions amongst its constituent networks.

6. Summary and conclusions

Modeling of the Alkanoamine regeneration tower, using the NCENN technique developed in this work, can be useful for the plant analysis, i.e., for gaining a better understanding of the plant. Other applications of the technique include: monitoring of the process, designing controllers, plant optimization, fault detection and component diagnosis. NCENN utilizes GA as the primal learner algorithm and then employs LMA with a negative correlation objective function as the supplement learner algorithm to identify a regression type of task. It was shown that due to their limitations, none of these two algorithms can exclusively model the process properly. Using a negative correlation objective function permits the component networks to be trained simultaneously whilst the network errors are correlated negatively. Consequently, GA is employed again to enhance the identification accuracy by assigning a weight to each of the correlated networks. Finally, the weighted averaging combination of the outputs of the component networks generates the NCENN output. As shown in this paper, the proposed NCENN outperformed the bagging and SNN methods in terms of modeling accuracy. This improvement is achieved through a reasonable increase in the computational load.

Appendix A

The MATLAB R2010b software is used for implementation of the considered algorithm used in the Manuscript. We developed programs for Neural Networks and Genetic Algorithm to model an actual chemical plant.

Actual data file (GTP.mat) has been recorded for the Regenerator Tower in the Gas Treatment Plant (GTP) of the “AMMAK” project in the Ahwaz onshore field of Iran.

Following code is “main” file of our program which each parameter are explained according body text of Manuscript for interested researchers.

```
clc; clear;
global x y M t tt bias1 bias2 Yresult ybar lambda;
run data;   % loading data from data.m file in Equation (1) and (2) in body
text
m=mean(output);s=std(output);    % data pre-processing
output = (output-mean(output))/std(output);% loading output data from
data.m
D  = size(input , 1);% number of all data set
input = (input-ones(D,1)*mean(input))./(ones(D,1)*std(input));% loading
input data from data.m
t_percent=80; % 80% of the sub-data sets for training and 20% for testing
rr= rand(1,D); [~,rri]=sort(rr);%randomly selection sub-data sets for
training
x = input(sort(rri(1:round((t_percent/100)*D))),:);% input train data
separation
y = output(sort(rri(1:round((t_percent/100)*D))));% output train data
separation
T=size(y,1);  % number of train data set
xs=size(x,2);  % number of inputs
J=12; % number of hidden layer neurons
M=10; % Ensemble Neural Network using 10 correlated and diverse component
neural networks is used to identify
lambda = 0.5;    % parameter ? adjusts the strength of the penalty term
t=ones(J,1)*[1:xs]+[0:J-1]'*ones(1,xs)*xs;
bias1=(ones(T,1)*((xs*J+1):((xs+1)*J)))'; % bias entered to hidden layer
tt=((xs+1)*J+1):((xs+2)*J);
bias2=J*(xs+2)+1; %bias entered to output layer
num =J*(xs+2)+1; % Length of individuals in GA
Yresult=zeros(T,M); %creating zero matrices
weights=zeros(M,num);%creating zero matrices
fcn  = @errorfcn; % objective function used in Equation (3) in body text
for combination
fcn2=@weight; %  objective function used in Equation (8) in body text for
combination
counter = 1;Error = 1;Errornew = 1;
for ii = 1:M
    weights(ii,:)= ga(fcn,num,[],[],[],[],-ones(1,M),ones(1,M)); %employing
GA to estimate the primal weights which directs the objective
    % function to its global minimum point
while counter<500 && Errornew<=Error % 500 epochs for the training
procedure
    Error = Errornew;
    counter = counter+1;
    for ii = 1:M
        Yresult(:,ii)=snn(weights(ii,:));%output of the i-th network
    end
    ybar= sum(Yresult ,2)/M; %the average of all component network outputs
obtained by Equation (6) in body text
    for ii =1:M
    weights(ii,:) = levenberg_marquardt(weights(ii,:),ii);clc %use LMA to
precisely tune the weights (W_i) using
    %the negative correlation objective function repesented in Equation
(4)in body text
    end
   Errornew = mean((y-ybar).^2);% stopping Criteria
end
```

```
   ww = ga(fcn2,M,[],[],[],[],zeros(1,M),ones(1,M)); % weight,Wi',  assigned
to each of the trained CNNs outputs using GA
   yresult=Yresult*(ww')/sum(ww);%Using the weighted averaging, the final
prediction of the output matrix of
   %NCENN using Equation (11) in body text
   cc = corrcoef(y*ones(1,M)-Yresult);%The correlation coefficient parameter
between the k-th and
   % j-th networks given by Equation (13) in body text
   titr = ['MSE for Train data   ',num2str((mean(((yresult-y).^2))))]; %titr
of result figures of trainig
   MSEtrain=(mean(((yresult-y).^2)))%To evaluate the performances of the
algorithms developed in the paper and
   %compare them with other methods or architectures used mean square
error(MSE)by Equation (19) in body text
   y = y*s+m;yresult=yresult*s+m;%denormalize output train data to their
normal units
   figure(1);hold off ;plot(y);hold on;plot(yresult,'r'); %loading result
figures of Train data
   legend data result; title(titr) ; hold off; xlabel('Time (s) ');ylabel('H2S
Content (ppm)');% details of result figures of Train data

   %testing
   x = input(sort(rri(round((t_percent/100)*D):end)),:);% input test data
separation
   y = output(sort(rri(round((t_percent/100)*D):end)));% input train data
separation
   T=size(y,1);% number of test data set
   t=ones(J,1)*(1:xs)+(0:J-1)'*ones(1,xs)*xs;
   bias1=(ones(T,1)*((xs*J+1):((xs+1)*J)))';% bias entered to hidden layer
   tt=((xs+1)*J+1):((xs+2)*J);
   bias2=J*(xs+2)+1;% bias entered to output layer
   Yresult=zeros(T,M);
   for ii = 1: M
       Yresult(:,ii)=snn(weights(ii,:));
   end
   yresult=Yresult*(ww')/sum(ww);%Using the weighted averaging, the final
prediction of the output matrix of
   %NCENN using Equation (11) in body text
   titr = ['MSE for Test data=   ',num2str((mean(((yresult-y).^2))))];%titr of
result figures of testing
   MSEtest=(mean(((yresult-y).^2)))%To evaluate the performances of the
algorithms developed in the paper and
   %compare them with other methods or architectures used mean square
error(MSE)by Equation (19) in body text
   y = y*s+m;yresult=yresult*s+m;%denormalize output test data to their normal
units
   figure(2);plot(y);hold on;plot(yresult,'r');
   legend data result;title(titr);hold off;xlabel('Time (s)');ylabel('H2S
Content  (ppm)');% details of result figures of Train data
   MAEtest=mean(abs(yresult-y))%To evaluate the performances of the algorithms
developed in the paper and compare them with other methods or architectures
used mean absolute error(MAE)by Equation (20) in body text
   stdev=std((yresult-y))%To evaluate the performances of the algorithms
developed in the paper and
   %compare them with other methods or architectures used standard deviation
(STD)by Equation (21) in body text
```

References

Abedinzadegan, M., 2008. Design and operations of natural gas sweetening facilities. In: Course prepared for National Iranian Gas Company Workshop for the 2nd Iranian Gas Forum.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. J. Mach. Learn. 36 (1), 105–139.

Breiman, L., 1996. Bagging predictors. J. Mach. Learn. 24 (2), 123–140.

Cunningham, P., Carney, J., Jacob, S., 2000. Stability problems with artificial neural networks and the ensemble solution. J. Artif. Intell. Med. 20 (3), 217–225.

Drucker, H., Cortes, C., Jackel, L.D., LeCun, Y., Vapnik, V., 1994. Boosting and other ensemble methods. J. Neural Comput. 6, 1289–1301.

Efron, B., 1979. Bootstrap methods: another look at the Jackknife. J. Ann. Stat. 7 (1), 1–26.

Fayazi, A., Arabloo, M., Shokrollahi, A., Zargari, M.H., Ghazanfari, M.H., 2014. State-of-the-art least square support vector machine application for accurate determination of natural gas viscosity. J. Indust. Eng. Chem. Res. 53, 945–958.

Fortuny, M., Baeza, J.A., Gamisans, X., Casas, C., Lafuente, J., Deshusses, M.A., Gabriel, D., 2008. Biological sweetening of energy gases mimics in biotrickling filters. J. Chemosph. 7, 10–17.

Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, San Francisco, USA.

Hansen, L.K., Salamon, P., 1990. Neural network ensembles. Trans. IEEE J. Pattern Anal. Mach. Intell. 12 (10), 993–1001.

Haykin, S., 1994. Neural Networks: a Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Isermann, R., Munchhof, M., 2009. Identification of Dynamic Systems. Springer.

Liu, Y., Yao, X., 1999. Ensemble learning via negative correlation. J. Neural Networks 12 (10), 1399–1404.

Maddox, R.N., Sheerar, L.F., 1994. Gas conditioning and processing. In: Campbell Petroleum Series, third ed., 4

Seqatoleslami, N., Koolivand Salooki, M., Mohammadi, N., 2011. A neural network for the gas sweetening absorption column using genetic algorithm. J. Petrol. Sci. Technol. 29, 1437–1448.

Shafiei, E., Jazayeri-Rad, H., 2012. Improving the identification performance of an industrial process using multiple neural networks. J. Intell. Syst. 4 (2), 40–44.

Sharkey, A.J.C., 1996. On combining artificial neural nets. J. Connect. Sci. 8, 299–313.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. Syst. Man Cybernet. 22 (3), 418–435.

Yao, H.M., Vuthaluru, H.B., Tade, M.O., Djukanovic, D., 2005. Artificial neural network based prediction of hydrogen content of coal in power station boilers. J. Fuel 84, 1535–1542.

Zio, E., Baraldi, P., Gola, G., 2008. Feature-based classifier ensembles for diagnosing multiple faults in rotating machinery. J. Appl. Soft Comput. 8 (4), 1365–1380.

Nomenclature

Greeks

λ: adjustor the strength of penalty term
β^2: variance of experimental data

Symbols and functions

X: input experimental data
$F(X)$: output experimental data
Z: the size of data set
M: number of individual neural network
W_i: weight learned by LMA
$f_i(x)$: output of ith network
$E_{iGA(x)}$: objective function used to learn primal weights by GA
$E_{iLMA(x)}$: objective function used to learn precise weights by LMA
$P_i(x)$: penalty function
$\overline{f(x)}$: average of networks output
W_i: weight assigned to each network
$E_{Ens.}$: objective function used to learn precise weights by LMA
$f_{pred}(x)$: final prediction output matrix
$f_{pred,j}(x)$: jth predicted data
R: actual and predicted output difference
$Corr_{kj}$: correlation coefficient
C_{kj}: covariance of kth and jth network
C_{kk}: variances for the network kth
C_{jj}: variances for the network jth
t: time

Abbreviations

ANN: Artificial Neural Network
SNN: Single Neural Network
ENN: ensemble neural network
NCENN: negative correlation ensemble neural network
NCL: negative correlation learning
GA: Genetic Algorithm
LMA: Levenberg–Marquardt Algorithm
MEA: MonoEthanolAmin
DEA: DiEthanolAmin
MDEA: MethylDiEthanolAmin
GTP: gas treatment plant
MSE: mean square error
MAE: mean absolute error
STD: standard deviation