International Journal of Computational Intelligence Systems, Vol.3, No. 5 (October, 2010), 622-631

Extremal Optimization Combined with LM Gradient Search for MLP Network Learning

PENG CHEN *
Department of Automation, Shanghai Jiaotong University
Shanghai, 200240, P. R. China
E-mail: pengchen@sjtu.edu.cn

YONG-ZAI LU
Department of Automation, Shanghai Jiaotong University
Shanghai, 200240, P. R. China
E-mail: y.lu@ieee.org

YU-WANG CHEN
Manchester Business School, The University of Manchester
Manchester M15 6PB, UK
E-mail: yu-wang.chen@mbs.ac.uk

Received: 16-10-2009
Accepted: 13-08-2010

Abstract

Gradient search based neural network training algorithm may suffer from local optimum, poor generalization and slow convergence. In this study, a novel Memetic Algorithm based hybrid method with the integration of "extremal optimization" and "Levenberg–Marquardt" is proposed to train multilayer perceptron (MLP) networks. Inheriting the advantages of the two approaches, the proposed "EO-LM" method can avoid local minima and improve MLP network learning performance in generalization capability and computation efficiency. The experimental tests on two benchmark problems and an application example for the end-point-prediction of basic oxygen furnace in steelmaking show the effectiveness of the proposed EO-LM algorithm.

Keywords: Back propagation; Extremal optimization; "Levenberg–Marquardt" (LM) gradient search; Memetic Algorithms; Supervised learning;

1. Introduction

The properties of feed-forward multilayer perceptron (MLP) network are governed by the activation functions of neurons and the synaptic connections between the layered neurons, as shown in Fig. 1. The associative memories from input space to output space are built up and stored in the synaptic weights through supervised learning from learning examples. The performance under its working environment measures the generalization capability of a MLP network.[1] After introduced by Werbos[2] and popularized by Rumelhart,[3,4] the gradient search (GS) based Back Propagation (BP) algorithm has been the most popular learning technique in MLP network training due to its implementation simplicity and applicability. However, in view of the drawbacks of gradient search in nature, such as easily trapping into local minima, sensitivity to initial weights and poor generalization, etc.[1,5] there have been a variety of well

* Corresponding author. Email: pengchen@sjtu.edu.cn.

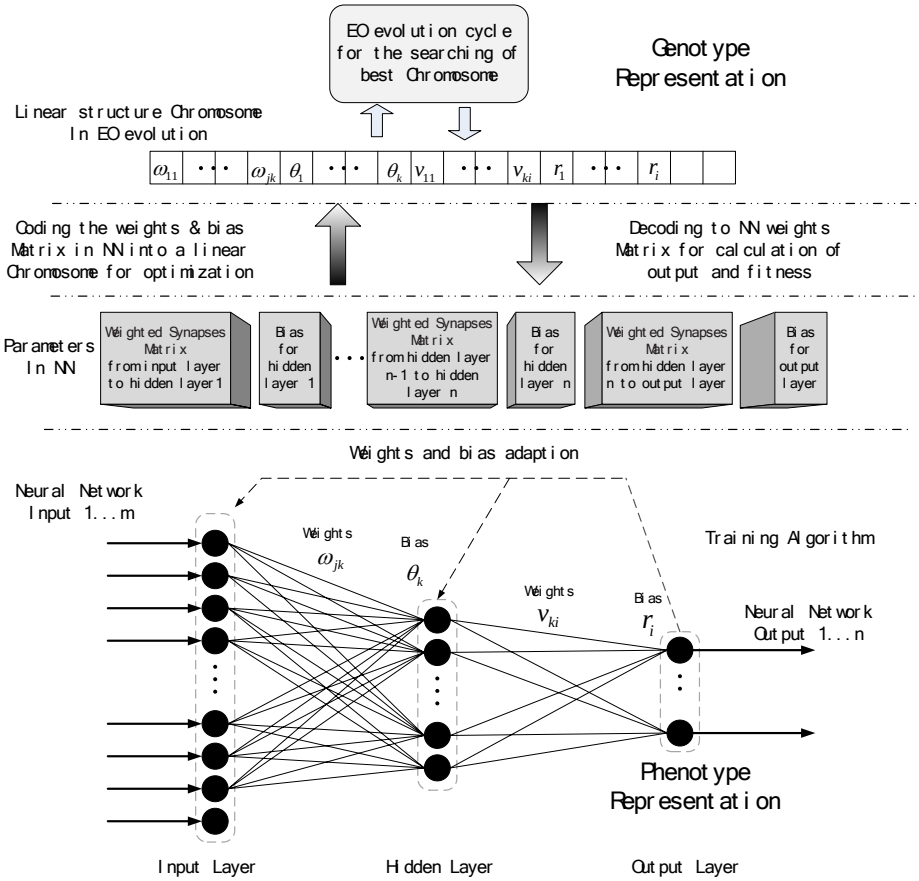


Fig. 1. Mapping neural network weights/biases from Phenotype space into a linear chromosome of Genotype space

known attempts to improve the original BP algorithm.[6-8] The applications of these approaches may result in better solutions, but require higher computation cost.[9]

On the other hand, Hush has proved that the parameter optimization for a MLP network with sigmoid function is a NP-Hard problem.[10] The recent research results in bio-inspired computational intelligence[11] (e.g., evolutionary algorithms, extremal optimization and ant colony optimization) and their superior capabilities in solving NP-hard and complex optimization problems have motivated researchers to use computational intelligence methods for the training of MLP network. One way to overcome the drawbacks of the BP learning is to formulate the training process as computational intelligence based evolution of MLP network structure, synaptic weights, learning rule and input features, etc.[9,12-17] In fact, the NN evolution with computational intelligence methods may significantly enlarge its search space and provide better performance than BP algorithms. However, most computational intelligence methods are rather inefficient in fine-tuned local search although they are good at global search, especially when the searching solutions approach to a local region near the global optimum, this will result in high computation cost. Moreover, a particular class of global-local search hybrids with both efficiency and robustness named "Memetic Algorithms" (MAs) have been proposed in recent years, which are motivated by Richard Dawkins's concept of a meme representing a unit of cultural evolution that can exhibit local refinement.[18] MAs are a class of stochastic heuristics for global optimization which combine the global search nature of computational intelligence methods with local search to improve individual solutions.[19] In MAs, the rules governing global/local search are co-evolved and self-adapted alongside the problem representation within a coupled gene-meme evolutionary system. According to the No-Free-Lunch theorem,[20] a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge they incorporate, this fact clearly underpins the exploitation of problem knowledge intrinsic to MAs. The MAs have

been successfully applied to hundreds of real-world problems such as graph coloring,[21] vehicle routing problem,[22] nurse rostering problem[23] and bioinformatics,[24] etc. Consequently, the MA based hybrid optimization solutions are also applicable for the improvement of NN learning.

Based on the complexity of nonlinear optimization involved in NN learning, this study presents the development of a novel MA based hybrid method called "EO-LM" learning algorithm, which combines the recently proposed heuristic extremal optimization (EO)[25] with the popular Levenberg–Marquardt (LM) gradient search algorithm.[26] The rest of the paper is organized as follows: Section II gives the math formulation for the problem under study; Section III illustrates the EO-LM fundamentals and algorithms; Section IV shows the comparison results between EO-LM and standard LM algorithms on two experimental problems. Section V presents the industrial application of EO-LM learning for BOF end-point-quality prediction. The concluding remarks and future work are included in Section VI.

2. Problem Statement and Math Formulation

A feed-forward MLP network with a single hidden layer is shown in Fig. 1, if we select the tan-sigmoid and linear function as the activation functions of hidden and output layers respectively, the map from *j*th input x_j, $(j = 1...m)$ to *i*th output $\hat{y}_i$, $(i = 1...n)$ can be written as:

$$\hat{y}_i = f(X, w, v, \theta, r) = \sum_{k=1}^{p} \left(v_{ki} z_k + r_i \right) = \sum_{k=1}^{p} \left(v_{ki} \log\left(\sum_{j=1}^{m} \omega_{jk} x_j + \theta_k \right) + r_i \right) \quad (1)$$

$$i = 1...n.$$

z_k the *k*th hidden layer variable ($k = 1,...p$);

ω_{jk} the weight linking the *j*th input variable with the *k*th hidden layer variable;

v_{ki} the weight linking the *k*th hidden layer variable with the *i*th output variable;

θ_k the bias of the *k*th hidden layer variable;

r_i the bias of the *i*th output variable;

log sig the logistic transfer function

log sig (a)=1/[(1+exp(-a))]

In NN training, the learning samples are often divided into a training dataset and a validation dataset, the former is used for updating the network weights and biases. The error on the validation set is monitored during the training process, which will guarantee the generalization of the NN. The aim of this study is to develop a novel MA based hybrid approach to optimize the synaptic weights for a MLP network that may provide good performance in generalization and robustness with the minimum output error:

$$\begin{cases} \min E(w, v, \theta, r) = \sum_{i=1}^{n} \sum_{l=1}^{n\_Train} \left[y_i^l - \hat{y}_i^l \right]^2 \\ s.t \quad w \in R^{m*p}, v \in R^{p*n}, \theta \in R^p, r \in R^n \end{cases} . \quad (2)$$

where n_Train represents the training data number. w, v, θ, r are bounded by the searching space of the optimization algorithm. y_i represents the *i*th desired output.

3. EO-LM algorithm for MLP network Training

3.1. *Extremal Optimization*

The "Extremal Optimization (EO)" proposed by Boettcher and Percus[25] is derived from the fundamentals of statistical physics and self-organized criticality (SOC)[27] based on the Bak-Sneppen (BS) model[28] which simulates far-from equilibrium dynamics in statistical physics. SOC states that large interactive systems evolve to a state where even a minor change in a single element may lead to generating avalanches or domino effects that can reach any other element in the system. In contrast to other evolutionary computational methods which operate on an entire "gene-pool" of huge number of possible solutions, EO successively eliminates those worst components in the sub-optimal solutions. Its large fluctuations provide significant hill-climbing ability, which enables EO to perform well particularly at the phase transitions.[29] For an optimization problem with *n* decision variables, EO proceeds as follows:

(i) Initialize a configuration S at will, set $S_{best} = S$,

(ii) For the current solution S,

(a) Evaluate the fitness for each decision variable x_i,

(b) Rank all the components by their fitness and find the component with the "worst fitness",

(c) Choose one solution S' in the neighborhood of S, i.e., such that the worst component x_j must change its state,
(d) Accept $S = S'$ unconditionally,
(e) If $F(S) < F(S_{best})$, set $S_{best} = S$,
(iii) Repeat Step(2) as long as desired.
(iv) Return S_{best} and $F(S_{best})$.

Generally speaking, EO is particularly useful for dealing with large complex problems with rough landscape or multiple local optima. It is less likely to be trapped in local minima than traditional gradient-based search algorithms. Benefited from its generality and ability of exploring complicated configuration spaces, EO and its derivatives have been successfully applied to some combinatorial or numerical optimization problems, such as graph bi-partitioning,[29] TSP,[29] graph coloring,[30] spin glasses,[31] MAX-SAT[32] and dynamic combinatorial problems.[33] The research results by Chen and Lu show EO can be effectively applied in solving combinatory and multi-objective hard benchmarks and real-world optimization problems.[34-37]

3.2. *Levenberg–Marquardt algorithm*

The Levenberg-Marquardt (LM) gradient search algorithm was introduced to the feed-forward network training to provide better performance.[26] Generally, the LM algorithm is a Hessian-based algorithm for nonlinear least squares optimization.[38] Similar to the quasi-Newton methods, the LM algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. Under the assumption that the error function is some kind of squared sum, the Hessian matrix can be approximated as:

$$H = J^T J. \tag{3}$$

and the gradient can be computed as:

$$g = J^T e. \tag{4}$$

where J is the Jacobian matrix that contains first derivatives of the network errors with respect to weights and biases, and e is an error vector. The Jacobian matrix can be computed through a standard BP technique that is much less complex than computing the Hessian matrix.[26]

The LM algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e. \tag{5}$$

The parameter μ is a scalar controlling the behavior of the algorithm. The convergence behavior of the LM is similar to that of the Gauss-Newton method. Near a solution with small residual, it performs well and gives a very fast convergence rate; while for the large-residual case, the performance of the Gauss-Newton and LM algorithms is usually poor.[38]

3.3. *Hybrid EO-LM algorithm and workflow*

As mentioned above, the efficiency of evolutionary training can be improved significantly by incorporating a local search procedure into the optimization; the local search algorithm could be gradient based methods such as LM or other methods. In this study, a hybrid EO-LM algorithm is developed and applied in NN network training. The structure of the new algorithm is based on the standard EO, the characteristic of the gradient search is added by propagating the individual solution with LM algorithm during EO evolution. The proposed EO-LM solution has the abilities in avoiding local minimum and performing the detailed local search with both efficiency and robustness. The incorporation of stochastic EO method with the conventional deterministic LM algorithm can combine the global explorative power of EO with the local exploitation behaviors of LM, complementing their individual weak points, and thus make MLP network training superior in generalization, computation efficiency and avoiding local minima. The EO-LM learning is executed between two phases in parallel: the genotype phase for EO-LM and the phenotype phase for NN.

Here we illustrate workflow of the algorithm and introduce three mutation operators adopted in this paper: the standard EO mutation, LM mutation and Multi-start Gaussian mutation. To utilize the advantages of each mutation operator, one or more phases of local search (mutation operator) are applied to the best solution S so far based on a probability parameter p_m in each generation. In contrast to the standard EO mutation, when LM mutation or Multi-start Gaussian mutation is adopted, we use the "GEO_{var}"[39] strategy to evolve the current solution by improving all variables simultaneously, as an attempt to speed up the process of searching the local minimum. There are two evolutionary levels during the proposed EO-LM optimization: On one hand evolution takes place at the "chromosome level" as in any other Evolutionary Algorithm; chromosomes (genes) represent solutions

and features of the problem one is trying to solve. On the other hand, evolution also happens at the "meme level", that is, the behaviors that individuals will use to alter the survival value of their chromosomes.[40] Accordingly, the solutions are evaluated by fitness functions of two different levels: The fitness of the respective gene itself (global fitness) and the interaction fitness between the respective gene and the respective meme (local fitness). Thus, both genetic and meme materials are co-evolved, the evolutionary changes at the gene level are expected to influence the evolution at the meme level, and vice versa. The proposed EO-LM is able to self-assemble different mutation operators and co-evolve the behaviors it needs to successfully solve the NN supervised learning problem. The flowchart of the proposed EO-LM algorithm to optimize parameters (the connection weights and the biases) of MLP network is shown in Fig. 2.

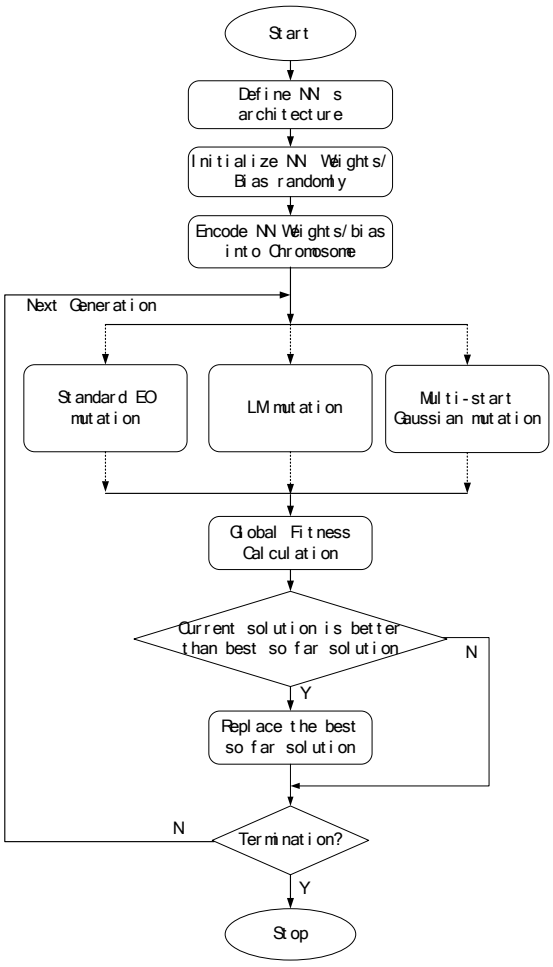


Fig. 2. Flowchart of the EO-LM algorithm

The work steps of the proposed EO-LM based MLP training algorithm in this study can be described below:

(i) Define the number of hidden layer, the numbers of input neurons, output neurons and the control parameters to be used in EO-LM algorithm.

(ii) Initialize the neural network with randomly generated weights and biases based on the predefined structure in step-i.

(iii) Map the weights/biases matrices of the neural network from the problem oriented phenotype space into a chromosome, as shown in Fig. 1.

(iv) For the first iteration of EO, decode the initial chromosome S back to weights/biases matrices and calculate the object fitness function, set $S_{best} = S$.

(v) Decide what kind of mutation operators should be imposed to the current chromosome S based on randomly generated probability parameters p_m , if $P_m \le P_{m\_basic}$, goes a); if $P_{m\_basic} < P_m \le P_{m\_LM}$, goes b); else if $P_m > P_{m\_LM}$, goes c)

(a) Perform the standard EO mutation on the best so far solution S .

(1) Change the value of each component in the current S and get a set of new solutions $S'_k, k \in [1,2,...,n]$.

(2) Sequentially evaluate the localized fitness λ_k specified in Eq. (10) for every S'_k , and rank them according to their fitness values.

(3) Choose the best solution S' from the new solutions set $[S']$, which is a neighbor subspace of the best so far solution S .

(b) Perform the LM mutation on the current chromosome S .

(1) Decode the chromosome S back to weights/biases matrices in MLP networks.

(2) The weight vector is updated for *N* iterations by:

$$S' = S - [J^T J + \mu I]^{-1} J^T e. \qquad (6)$$

where *J* is the Jacobian matrix, *e* is a vector of network errors defined in Eq. (7)

$$e(x) = \sum_{i=1}^{n} \sum_{l=1}^{n\_Train} (y_i^l - \hat{y}_i^l)^2. \qquad (7)$$

(3) Encode the updated weights/biases matrices to the chromosome S' .

(c) Perform the Multi-start Gaussian mutation on the current chromosome S. Multi-start methods have their main objective to increase diversity, whereby larger parts of the search space are explored.[11] This strategy is often adopted in MA to explore the neighborhood of the current solution.[19]

(1) Generate a new chromosome S_0' by adding a Gaussian distribution random vector with n dimensional to the best so far chromosome S.

$$S_0' = S + Scale * N(0,1). \quad (8)$$

where n is the length of chromosome, *Scale* is the mutation step size.

(2) Decode the chromosome S_0' back to weights/biases matrices in MLP networks.

(3) Training the MLP network based on LM algorithm in Eq. (6) for M iterations.

(4) Encode the updated weights/biases matrices to the chromosome S'.

(vi) Decode the chromosome S' back to weights/biases matrices and calculate the global object fitness function. If $F(S') < F(S_{best})$, Set $S_{best} = S'$.

(vii) If the termination criteria are not satisfied, go to step-v, else go next.

(viii) Return S_{best}.

3.4. *Fitness function*

The fitness function measures how fit an individual (i.e., solution) is, and the "fittest" one has more chance to be inherited into the next generation. A "global fitness" must be defined to evaluate how good a solution is. The errors on training set and validation set are often used to control and monitor the NN training process. Over-fitting usually occurs during the NN training with descending training error and ascending prediction error. It greatly debilitates the generalization ability of a network. Consequently, in this paper, the global fitness is defined as the sum of root mean square error (*RMSE*) on training set (*LRMSE*) and validation set (*GRMSE*), as defined in Eq. (9):

$$Fitness_{global}(S) = LRMSE_{S(w,v,\theta,r)} + GRMSE_{S(w,v,\theta,r)} = \sqrt{\frac{\sum_{i=1}^{n}\sum_{l=1}^{n\_Train}(y_i^l - \hat{y}_i^l)^2}{n * n\_Train}} + \sqrt{\frac{\sum_{i=1}^{n}\sum_{l=1}^{n\_Valid}(y_i^l - \hat{y}_i^l)^2}{n * n\_Valid}}. \quad (9)$$

Unlike GA, which works with a population of candidate solutions, EO depends on a single individual (i.e. chromosome) based evolution. Through performing mutation on the worst component and its neighbors successively, the individual in EO can evolve itself towards the global optimal solution generation by generation. This requires a suitable representation which permits each component to be assigned a quality measure (i.e. fitness) called "local fitness". In this paper, the local fitness λ_k is defined as an improvement in *LRMSE* made by the mutation imposed on the kth component of best so far chromosome S:

$$\lambda_k = Fitness_{local}(k) = \Delta LRMSE(k) = LRMSE_{S(w,v,\theta,r)} - LRMSE_{S_k'(w,v,\theta,r)}. \quad (10)$$

As long as the neural network is not over-fitting, the improvement on the local fitness λ_k will also improve the global fitness described in Eq. (9).

4. Experimental tests on Benchmark problems

This section presents the experimental tests of EO-LM algorithm in two benchmark problems. Without loss of generality, input/output data used are normalized to the range [0.1, 0.9].

4.1. *Example 1: A MISO (multi-input, single-output) static nonlinear function*

In this example, the EO-LM algorithm is applied to establish a MISO map specified by a highly nonlinear function as follows:

$$Y = 0.2(x_1^{0.5} + 2x_1x_2^{0.5} + x_2x_3 + x_3). \quad (11)$$

First we randomly generate 300 I/O observation pairs, for which the input variables are generated randomly in the regions I: $x_1 \in [0,1]$, $x_2 \in [0,1]$, $x_3 \in [0,1]$, use 200 of them as learning data and the other 100 as interpolation test data to be used for measuring the generalization performance, we randomly generate another 100 I/O data pairs within the regions II: $x_1 \in [1,2]$, $x_2 \in [1,2]$, $x_3 \in [1,2]$, they are beyond the region I and will be used as an extrapolation test data set. In this case, the structure of the network was {3, 2, 1}, namely three, two and one nodes in input, hidden and output layers, respectively, totally 11 parameters need to be optimized in this example.

For the purpose of fair comparison, this test was repeated 10 times using a Monte Carlo method for EO-

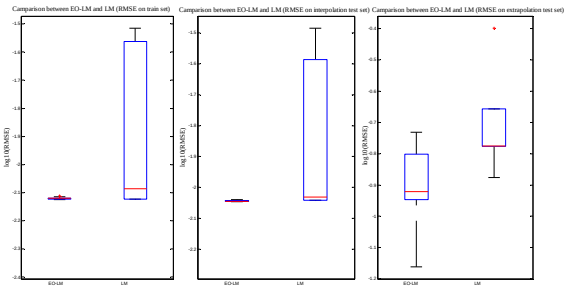


Fig. 3. The comparison of RMSE distribution between EO-LM and LM on train/ interpolation test/ extrapolation test set

LM and standard LM algorithms. The comparison results between these two algorithms are listed in Table 1. In addition to the popular criteria of root mean square error (RMSE) and mean error (ME), the efficiency coefficients R^2 or R that measures the proportion of the variation of the observations around the mean usually explained by the fitted regression model is also be used as an additional measure. The value of R^2 or R falls between 0 and 1. When the R^2 or R reaches to 1, the model's outputs perfectly agree with its system's actual outputs. The Table 1 and Fig. 3 show the comparison between EO-LM and LM based on the statistical data.

Table 1 shows the performance of each algorithm over the 10 runs, giving an indication of the robustness and the generalization ability of each algorithm. As mentioned above, the better solution has smaller RMSE over "train", "interpolation test" and "extrapolation test" datasets, and lower standard deviation as well. We can see that the EO-LM algorithm performs much better than standard LM algorithm. Fig. 3 shows comparison of RMSE distribution between EO-LM and LM on training and test data. From Table 1 and Fig. 3, we can see that the solution is more robust and consistent than the solution evolved by standard LM.

Fig. 4 gives the comparisons in generalization performance between EO-LM and LM algorithms on training, interpolation and extrapolation test datasets. We can see both algorithms provide good results for

Table 1. Comparison between the EO-LM and LM

		EO-LM ALGORITHM			LM ALGORITHM		
		Train set	**Test data set (interpolation)**	**Test data set (extrapolation)**	**Train set**	**Test data set (interpolation)**	**Test data set (extrapolation)**
RMSE	**Max**	0.0077	0.0091	0.1857	0.0306	0.0328	0.4000
	Min	0.0075	0.0090	0.0688	0.0075	0.0091	0.1328
	Mean	0.0076	0.0090	0.1273	0.0141	0.0149	0.2212
	Std Dev	6.76e-05	4.56e-05	0.0378	0.00997	0.0094	0.0977
R^2	**Mean**	0.9985	0.9970	0.7819	0.9924	0.9890	N/A
Training Time		61.44s			74.53s		

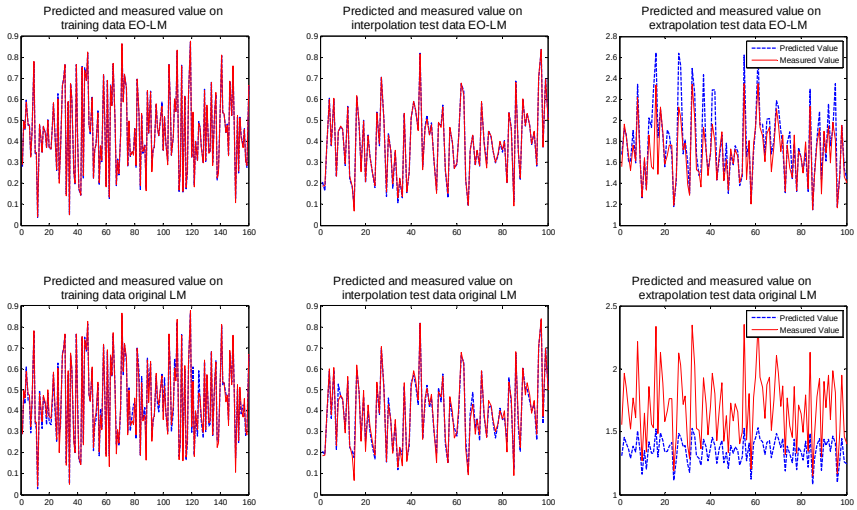


Fig. 4. The performance comparison between EO-LM and LM on train/ interpolation test/ extrapolation test set

training dataset and EO-LM performs slightly better for the interpolation test dataset, but much better for the extrapolation test dataset. The extrapolation test dataset is a most crucial test phase to tell which algorithm will have a better ability to learn, EO-LM performs pretty well in this case, while LM corrupts.

4.2. *Example 2: dynamic modeling for continuous stirred tank reactor (CSTR)*

The system considered here is an isothermal CSTR process with first order reaction $A+B\rightarrow P$, in the presence of excess concentration of A. The corresponding MIMO CSTR model is as following[41]:

$$\begin{cases} \dfrac{dx_1}{dt} = u_1 + u_2 - k_1\sqrt{x_1} \\ \dfrac{dx_2}{dt} = (C_{B1} - x_2)\dfrac{u_1}{x_1} + (C_{B2} - x_2)\dfrac{u_2}{x_1} - \dfrac{k_2 x_2}{(1+x_2)^2} \end{cases} \quad (12)$$

$$y = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T .$$

$$\text{s.t} \quad \begin{cases} 0.1 \le u_1 \le 2 \\ 0.1 \le u_2 \le 2 \end{cases} \quad (13)$$

where $k_1 = 0.2,\ k_2 = 1,\ C_{B1} = 24.9,\ C_{B2} = 0.1$

The plant has two inputs and two outputs and their relationship expresses strong nonlinearity. For $(u_1, u_2) = (1,1)$, the CSTR has three equilibrium points at $x_1 = 100$, $x_2 = (0.633, 2.72, 7.07)$, with the middle equilibrium point being unstable, and the others stable.

Here we use the nonlinear autoregressive model with Exogenous inputs (NARX) to model the dynamic systems shown in Eq. (12) and Eq. (13), it can be mathematically represented as follows:

$$Y(n) = f\left(u_1(n-1), u_2(n-1), y_1(n-1), y_2(n-1)\right) \quad (14)$$

where $Y(n) = [y_1(n), y_2(n)]^T = [x_1(n), x_2(n)]^T$, In this case, a multi-input multi-output (MIMO) MLP network is employed to build the nonlinear mapping $f(\bullet)$ between inputs and outputs. The structure of the network was {4, 10, 2}, totally 72 parameters need to be optimized. We randomly generate 500 observation pairs and use 400 of them as the learning data and the other 100 as the test data set. The example was run 10 times for each algorithm randomly. The performance by EO-LM and standard LM is listed in Table 2.

Table 2 shows the comparison between predicted and measured values at training and test phases by hybrid EO-LM and LM. We can see that EO-LM performs better than standard LM both on train data and test data.

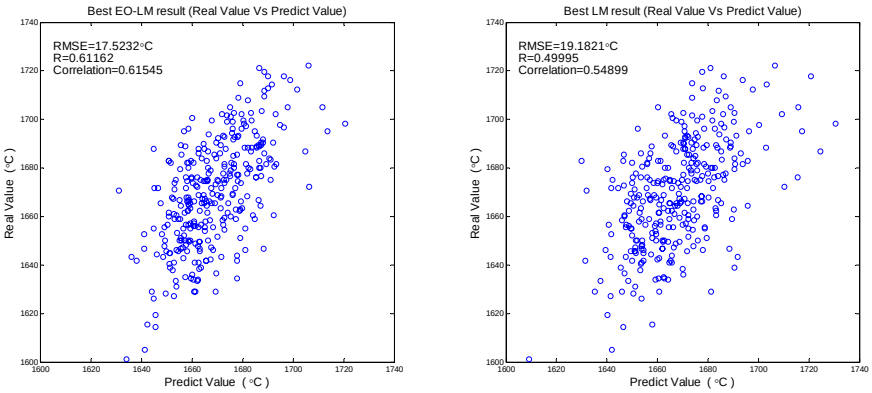
5. Application to end-point temperature prediction model of a production scale BOF

In this section, the proposed hybrid EO-LM algorithm is applied to a practical engineering problem for a production scale batch Basic Oxygen Furnace (BOF) in steelmaking. BOF is a complex multivariable and multiphase chemical reactor to produce the desired steel grade. The precise prediction of the end-point-temperature as a key performance index is vital for BOF operation. To evaluate the effectiveness of the proposed hybrid EO-LM for the predictions of end-point temperature, we perform the simulation experiment using real industry data. First, we gather over 1600 data from the historical database of the steel plant. Among them, randomly select 800 pairs as training data, 480 pairs as validation data and the rest 320 pairs as test data. Subsequently, two learning algorithms are used to train the MLP networks. One is the proposed EO-LM method and the other is the conventional LM algorithm, the example was run 10 times for each algorithm randomly. Based on the mechanism of BOF process, a network {13, 10, 1} is used for the industrial tests. The performance comparisons between EO-LM and LM after 10 runs are listed in Table 3 and Fig. 5.

It can be seen the generalization performance of EO-LM evolved model is better than that of LM. The prediction RMSE of best model by EO-LM on test dataset is reduced by 8.6% compared with LM algorithm. The experimental results indicate that MA

Table 2. Comparison between the EO-LM and LM on CSTR Model

		EO-LM ALGORITHM				LM ALGORITHM			
		Train set		Test set		Train set		Test data set	
		Y1	Y2	Y1	Y2	Y1	Y2	Y1	Y2
RMSE	Best	0.0023	0.0024	0.0043	0.0045	0.0025	0.0031	0.0048	0.0048
	Mean	0.0024	0.0025	0.0046	0.0052	0.0027	0.0031	0.0051	0.0054
	Std Dev	8.05e-05	1.81e-04	2.41e-04	5.98e-04	3.01e-04	3.38e-04	3.53e-04	4.13e-04



and LM for BOF end-point temperature prediction

Table 3. Comparison between the EO-LM and LM on BOF end-point temperature prediction

	TEST SET RMSE			
	Mean	**Max**	**Min**	**Std Dev**
EO-LM	18.4344	19.1793	17.5232	0.5721
LM	19.6829	20.3726	19.1821	0.4156

based hybrid EO-LM algorithm proposed in this paper can easily avoid the local minima, over-fitting or under-fitting problems suffered by traditional GS based neural network learning algorithms.

6. Concluding Remarks

In view of the drawbacks of gradient based NN learning algorithms, a novel hybrid "EO-LM" learning algorithm with the integration of EO and LM has been developed and evaluated with two benchmark problems and a practical engineering problem for a production scale batch Basic Oxygen Furnace (BOF) in steelmaking. The main advantages of the proposed algorithm are to utilize the superior features of EO and LM in global and local search respectively. As a result, the applications of the proposed EO-LM learning algorithm in neural network training may create a neural network model with better performance, such as avoiding stuck in to local minima and having good generalization capability.

The future studies involve the applications of EO and its derivatives in optimizing both the architecture and parameters of NN and more fundamental research on evolution dynamics for more benchmark and real-world problems.

References

1. S. Haykin, *Neural Networks: A Comprehensive Foundation*, (Macmillan, New York, 1994).
2. P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Ph.D. dissertation, Harvard University, 1974.
3. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*. (MIT Press, Cambridge, MA, 1986), pp. 318-362.
4. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back propagating errors, *Nature*. 323(9) (1986) 533-536.
5. R. Salomon, Evolutionary algorithms and gradient search: similarities and differences, *IEEE Trans Evol Comput*. 2 (2) (1998) 45-55.
6. R. A. Jacobs, Increased rates of convergence through learning rate adaptation, *Neural Networks*. 1(4) (1988) 295-307.
7. A. K. Rigler, J. M. Irvine and T. P. Vogl, Rescaling of variables in back propagation learning, *Neural Networks*. 4(2) (1991) 225-229.
8. Y. Fukuoka, H. Matsuki, H. Minamitani and A. Ishida, A modified back propagation method to avoid false local minima, *Neural Networks*. 11(6) (1998) 1059-1072.
9. B. Dengiz, C. Alabas-Uslu and O. Dengiz, A tabu search algorithm for the training of neural Networks, *J. Oper. Res. Soc.* 60 (2) (2009) 282-291.
10. D. R. Hush, Training a sigmoidal node is hard, *Neural Comp.* 11(5) (1999) 1249-1260.
11. A. P. Engelbrecht, *Computational Intelligence, An introduction*, 2nd ed. (John Wiley & Sons, New York, 2007).
12. J. Arifovic and R. Gencay, Using genetic algorithms to select architecture of a feedforward artificial neural network, *Phys A Stat Mech Appl*. 289 (2001) 574-594.
13. A. Fasih, C. J. Chedjou and K. Kyamakya, Cellular Neural Networks-Based Genetic Algorithm for Optimizing the Behavior of an Unstructured Robot, *Int. J. Comput. Intell. Syst.* 2 (2) (2009) 124-133.
14. A. Reyaz-Ahmed, Y. Q. Zhang, R. W. Harrison, Granular Decision Tree and Evolutionary Neural SVM for Protein

Secondary Structure Prediction, *Int. J. Comput. Intell. Syst*. 2 (4) (2009) 343-352.
15. A. Sedkia, D. Ouazar and E. El. Mazoudi, Evolving neural network using real coded genetic algorithm for daily rainfall-runoff forecasting, *Expert Sys Appl*. 36 (3) (2009) 4523-4527.
16. S. J. Li, Y. Li, Y. Liu and Y. F. Xu, A GA-based NN approach for makespan estimation, *Appl. Math. Comput.* 185 (2) (2007) 1003-1014.
17. X. Yao and Md. M. Islam, Evolving artificial neural network ensembles, *IEEE Comput. Intell. Mag*. 3(1) (2008) 31-42.
18. R. Dawkins, *The Selfish Gene*, (Oxford University Press, Oxford, 1976).
19. C. Cotta and P. Moscato, Memetic Algorithms, in *Handbook of Approximation Algorithms and Metaheuristics* (CRC press, 2007), 27.1–27.12.
20. D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, *IEEE Trans Evol Comput*. 1 (1) (1997) 67-82.
21. Z. Lü and J. K. Hao, A memetic algorithm for graph coloring, *Eur J Oper Res*. 203 (1) (2010) 241-250.
22. Y. Nagata, O. Bräysy and W. Dullaert, A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows, *Comp. Oper. Res*. 37 (4) (2010) 724-737.
23. E. Ozcan, self-generation and nurse rostering, *Lect. Notes Comput. Sci*. 3867 (2007) 85-104.
24. Z. Zhu, Y. S. Ong and J. M. Zurada, Identification of full and partial class relevant genes, *IEEE/ACM Trans. Comput. BioL. Bioinf*. 7 (2) (2010) 263-277.
25. S. Boettcher and A. G. Percus, Extremal Optimization: Methods derived from Co-Evolution, in *Proceedings of the Genetic and Evolutionary Computation Conference*. (1999), pp. 825-832.
26. M. T. Hagan and M. B. Menhaj, Training feedforward networks with the Marquardt algorithm, *IEEE Trans Neural Networks*. 5 (6) (1994) 989-993.
27. P. Bak, C. Tang and K. Wiesenfeld, Self-organized criticality: An explanation of the 1/f noise, *Phys. Rev. Lett*. 59 (8) (1987) 381-384.
28. P. Bak and K. Sneppen, Punctuated equilibrium and criticality in a simple model of evolution, *Phys. Rev. Lett*. 71 (24) (1993) 4083-4086.
29. S. Boettcher and A. G. Percus, Nature's way of optimizing, *Artif. Intel.* 119 (2000) 275-286.
30. S. Boettcher and A. G. Percus, Extremal optimization at the phase transition of the 3-coloring problem, *Phys. Rev. E Stat. Nonlinear Soft Matter Phys*. 69 (6) (2004) 066703.
31. S. Boettcher, Extremal Optimization for the Sherrington-Kirkpatrick Spin Glass, *Eur. Phys. J. B*. 46 (4) (2005) 501-505.
32. M. E. Menai and M. Batouche, Efficient Initial Solution to Extremal Optimization Algorithm for Weighted MAXSAT Problem, *Lect Notes Artif Intell*. 2718 (2003) 592-603.
33. I. Moser and T. Hendtlass, Solving problems with hidden dynamics - comparison of extremal optimization and ant colony system, in *Proceedings of 2006 IEEE Congress on Evolutionary Computation*. (2006), pp. 1248-1255.
34. Y. Z. Lu, M. R. Chen and Y. W. Chen, Studies on extremal optimization and its applications in solving real world optimization problems, in *Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence*. (2007), pp. 162-168.
35. M. R. Chen and Y. Z. Lu, A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization, *Eur J Oper Res*. 188(3) (2008) 637-651.
36. Y. W. Chen, Y. Z. Lu and G. K. Yang, Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling, *Int J Adv Manuf Technol*. 36 (2008) 959-968.
37. Y. W. Chen, Y. Z. Lu and P. Chen, Optimization with extremal dynamics for the traveling salesman problem, *Phys A Stat Mech Appl*. 385 (1) (2007) 115-123.
38. J. Nocedal, J. W. Stephen, *Numerical optimization*, (Springer, New York, 2006).
39. F. L. Sousa, V. Vlassov and F. M. Ramos, Generalized extremal optimization: An application in heat pipe design, *Appl. Math. Model*. 28 (10) (2004) 911-931.
40. N. Krasnogor and S. Gustafson, A study on the use of "self-generation" in memetic algorithms, *Natural Comput*. 3 (1) (2004) 53-76.
41. F. Martinsen, L. T. Biegler and B. A. Foss, A new optimization algorithm with application to nonlinear MPC, *J Process Control*. 14 (8) (2004) 853-865.