The Use of Evolutionary Algorithm in Training Neural Networks for Hematocrit Estimation

Hieu Trung Huynh and Yonggwan Won
Department of Computer Engineering, Chonnam National University
Republic of Korea

1. Introduction

Evolutionary algorithms (EAs) have recently been successfully applied in optimization problems and engineering disciplines. They can solve complex optimization problems without specialized information such as gradient or smoothness of objective functions. Although pure EAs such as genetic algorithm (GA), evolutionary strategies (ES) and evolutionary programming (EP) are easy to implement and offer fair performance in many applications, experimental results have shown that a variant of evolutionary method namely Differential Evolution (DE) has good convergence properties and outperforms other well known EAs (Ilonen et al., 2003). This variation was first introduced by Storn and Price (Storn & Price, 1997) and has an increasing interest as an optimization technique in recent years due to its achievement for a global minimum. It has several important differences from the traditional genetic optimization especially in the nature of the mutation, in which instead of taking a random perturbation, DE randomly selects a pair of individuals and computes the difference between their parameter vectors. This vector of difference is then added to the individual being mutated after multiplying by a constant. Another important difference is that the DE does not require the selection of parents based on fitness. Instead, fitness determines which children are kept for the next generation. Advantages of these approaches are shown in (Storn & Price, 1997).
Using DE for training neural networks was first introduced in (Masters & Land, 1997). It was reported that the DE algorithm is particularly suitable for training general regression neural networks (GRNN), and it outperforms other training methods such as gradient and Hessian on applications which have the presence of multiple local minima in the error space. Recently, the combination of the DE and other training algorithms has also been investigated. Subudhi and Jena (Subudhi & Jena, 2008) proposed a combination of DE and Levenberg Marquardt (LM) to train neural network for nonlinear system identification. It was shown that this combination can offer better identification results than neural networks trained by ordinary LM algorithm. More comprehensive studies for using DE in the training neural networks are presented in (Ilonen et al., 2003).
Although there are many network architectures proposed for different problems and applications, it was shown that single hidden-layer feedforward neural networks (SLFNs) can form boundaries with arbitrary shape and approximate any function with arbitrarily small error if the activation functions are chosen properly (Huang et al., 2000). An efficient training algorithm namely extreme learning machine (ELM) was proposed for SLFNs. It

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

analytically determines the output weights with random choice of input weights and hidden biases. This algorithm can obtain good performance with high learning speed in many applications. However, it often requires a large number of hidden units which takes longer time for responding to new patterns. Hybrid approaches which use DE algorithm were proposed to overcome this problem (Zhu et al., 2005; Hieu & Won, 2008). Zhu et al. (Zhu et al., 2005) introduced a method called evolutionary extreme learning machine (E-ELM) which takes advantages of both ELM and DE. In E-ELM, the input weights and hidden layer biases are determined by DE process and the output weights are determined by Moore-Penrose (MP) generalized inverse like ELM. Another improvement of ELM was shown in (Hieu & Won, 2008) which is called evolutionary least squares extreme learning machine (ELS-ELM). Unlike ELM and E-ELM, input weights and hidden layer biases in ELS-ELM are first initialized by a linear model, and then the optimization is performed by applying DE process. Experimental results showed its better performance in regression problems.
In this chapter, we have a space for introducing the use of DE process in training SLFNs for estimating hematocrit density which is an important factor for surgical procedures and hemodialysis, and is the most highly affecting factor influencing the accuracy of glucose measurements with a hand-held device that uses the whole blood. The input features of the networks are sampled from transduced current curves which are produced by the transfer of ions to an electrode. These ions are produced by the enzymatic reaction in glucose measurement using the electrochemical glucose biosensors. The networks are trained by hybrid algorithms which take advantages of DE process, linear model, and ELM to obtain good performance with compact architectures.

2. Neural Nnetworks and Extreme Learning Machine (ELM)

The artificial neural network has been vastly used in machine learning due to its ability to provide proper models for classes of problems that are difficult to handle by using classical parametric techniques. A typical neural network consists of layers with units which are also called neurons. It was shown that a single hidden layer feedforward neural network (SLFN) can approximate any function with arbitrarily small error if the activation function is chosen properly. The typical architecture of a SLFN is shown in Fig. 1. The *i*-th output corresponding to the *j*-th input pattern of a SLFN with *N* hidden units and *C* output units can be represented by

$$o_{ji}=\mathbf{h}_j \cdot \boldsymbol{\alpha}_i, \tag{1}$$

where $\boldsymbol{\alpha}_i=[a_{i1}\ a_{i2}\ \ldots\ a_{iN}]^T$ is the weight vector connecting from the hidden units to the *i*-th output unit, $\mathbf{h}_j=[f(\mathbf{w}_1 \cdot \mathbf{x}_j+b_1)\ f(\mathbf{w}_2 \cdot \mathbf{x}_j+b_2)\ \ldots\ f(\mathbf{w}_N \cdot \mathbf{x}_j+b_N)]^T$ is the output vector of the hidden layer corresponding to pattern $\mathbf{x}_j \in \mathbb{R}^d$, b_m is the bias of the *m*-th hidden unit, and $\mathbf{w}_m=[w_{m1}\ w_{m2}\ \ldots\ w_{md}]^T$ is the weight vector connecting from the input units to the *m*-th hidden unit. Note that $\mathbf{x} \cdot \mathbf{y} =<\mathbf{x},\mathbf{y}>$ is the inner product of two vectors **x** and **y**.
For *n* training patterns ($\mathbf{x}_j$, $\mathbf{t}_j$), $j=1,2,\ldots, n$, where $\mathbf{x}_j=[x_{j1}\ x_{j2}\ \ldots\ x_{jd}]^T$ and $\mathbf{t}_j=[t_{j1}\ t_{j2}\ \ldots\ t_{jC}]^T$ are the *j*-th input pattern and target respectively, the main goal of training process is to determine the network weights $\mathbf{w}_m$, $\boldsymbol{\alpha}_i$, and b_m so that they minimize the error function defined by

$$E=\sum_{j=1}^{n}\left(\mathbf{o}_j-\mathbf{t}_j\right)^2=\sum_{j=1}^{n}\sum_{i=1}^{C}(\mathbf{h}_j\cdot\boldsymbol{\alpha}_i-t_{ji})^2 . \tag{2}$$

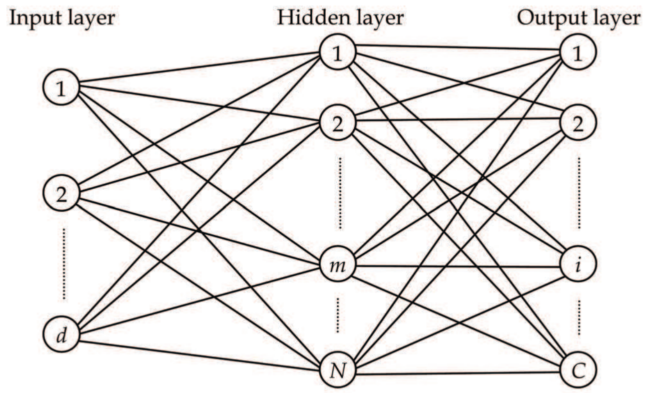

Fig. 1. Architecture of SLFN

Traditionally, this estimation of the network weights is performed based on the gradient-descent algorithms, in which the set of **G** vectors consisting of weights ($\mathbf{w}_i$, $\boldsymbol{\alpha}_i$) and biases b_i is iteratively adjusted as follows:

$$\mathbf{G}_k=\mathbf{G}_{k-1} - \eta \frac{\partial E}{\partial \mathbf{G}}, \tag{3}$$

where η is a learning rate. One of the popular methods based on the gradient descent is the back-propagation (BP) algorithm or generalized delta rule, in which gradients can be calculated and the parameter vector of the network can be determined by error propagation from the output to the input. However, the back-propagation algorithms are generally slow in learning due to improper learning parameters. They may also easily get over-fitting or be stuck in local minima. These problems have been overcome by many improvements proposed by many researchers. However, up to now, most of the training algorithms based on the gradient descent are still slow due to many learning steps which may be required in the learning processes.

Recently, a novel learning algorithm called extreme learning machine (ELM) was proposed for training SLFNs (Huang et al., 2006). The minimization process of error function in the ELM is performed by using a linear system:

$$\mathbf{HA}=\mathbf{T}, \tag{4}$$

where **H** is called as the hidden layer output matrix of the SLFN and defined by (Huang et al., 2006) :

$$\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \dots\ \mathbf{h}_n]^T = \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_N \cdot \mathbf{x}_1 + b_N) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_n + b_1) & \cdots & f(\mathbf{w}_N \cdot \mathbf{x}_n + b_N) \end{bmatrix}, \tag{5}$$

$$\mathbf{T}= [\mathbf{t}_1\ \mathbf{t}_2\ \dots\ \mathbf{t}_n]^T, \tag{6}$$

and

$$\mathbf{A}=[\,\alpha_1\ \alpha_2\ \ldots\ \alpha_C]. \tag{7}$$

The ELM algorithm was based on two basic principles:

1. If the activation function is infinitely differentiable in any interval of $\mathbb{R}$ and the number of training patterns equals the number of hidden units ($n=N$) then the hidden layer output matrix **H** is invertible. Based on this principle, it can randomly assign the input weights and biases of hidden units and the output weights are analytically calculated by simply inverting **H**.
2. When the number of hidden units is less than the number of input patterns ($N<n$), the output weight matrix **A** is also determined by using pseudo inverse of **H** with a small nonzero training error.

Thus, the extreme learning machine (ELM) can be described as follows:

Assign arbitrary input weights $\mathbf{w}_m$ and biases b_m, $m=1, 2, \ldots, N$.
Determine the output matrix **H** of the hidden layer by Eq. (5).
Determine the output weight matrix **A** as follows:

$$\hat{\mathbf{A}}=\mathbf{H}^{\dagger}\mathbf{T} \tag{8}$$

This algorithm can greatly reduce learning time for finding the input weights and biases of hidden units. However, it often requires more hidden units than conventional algorithms, and therefore application of the trained network to an input pattern takes longer time. This drawback is caused by existance of redundant hidden units. Therefore, it claims that the performance of SLFNs can be improved if the input weights and biases of hidden units are chosen properly.

3. Evolutionary algorithms for training single hidden layer feedforward neural networks

This section presents evolutionary computation based approaches for reducing the number of hidden units in ELM. The first approach proposed by Q.-Y. Zhu et al. (Zhu et al., 2005) was called evolutionary extreme learning machine (E-ELM). In E-ELM, a hybrid learning algorithm was proposed, in which the input weights and biases are determined by using the differential evolutionary algorithm and the output weights are analytically determined by using Moore-Penrose (MP) generalized inverse (Serre, 2002).

3.1 Hybrid of differential evolution and extreme learning machine

Similar to other evolutionary algorithms, DE operates on a population of candidate solutions called individuals. Each individual is a set of the input weights and biases of hidden units defined by $\boldsymbol{\theta} = [w_{11}, w_{12}, \ldots, w_{1d}, w_{21}, w_{22}, \ldots, w_{2d}, \ldots, w_{N1}, w_{N2}, \ldots, w_{Nd}, b_1, b_2, \ldots, b_N]$. DE always maintains NP individuals of population. At generation G, three steps of DE being mutation, crossover and selection are applied and individuals with better fitness values are retained to the next generation. The fitness of each individual is chosen as the root-mean squared error (RMSE) on the whole training set or the validation set. The output weights corresponding to each individual are determined by using the MP generalized inverse. In addition, in order to obtain smaller weight values, the norm of output weights $\|\mathbf{A}\|$ is added to the criterion of the selection step. We can summarize the E-ELM algorithm as follows:

- Generate the initial generation composed of parameter vectors { $\boldsymbol{\theta}_{i,0}$ | $i=1, 2, \ldots, NP$ } as the population.
- At each generation G, we do:

i. *Mutation:* the mutant vector is generated by $\mathbf{v}_{i,G+1}= \boldsymbol{\theta}_{r1,G}+F(\boldsymbol{\theta}_{r2,G} - \boldsymbol{\theta}_{r3,G})$, where $r1$, $r2$, and $r3$ are different random indices, and F is a constant factor used to control the amplification of the differential variation.

ii. *Crossover:* the trial vector is formed so that

$$\boldsymbol{\mu}_{ji,G+1} = \begin{cases} \mathbf{v}_{ji,G+1} \text{ if } rand\ b(j) \leq CR \text{ or } j = rnbr(i) \\ \boldsymbol{\theta}_{ji,G+1} \text{ if } rand\ b(j) > CR \text{ and } j \neq rnbr(i) \end{cases} \tag{9}$$

where *rand b(j)* is the *j*-th evaluation of a uniform random number generator, *CR* is the crossover constant and *rnbr(i)* is an index chosen randomly which ensures at least one parameter from $\mathbf{v}_{ji,G+1}$.

iii. *Determine the output weights.*

iv. *Evaluate the fitness for each individual.*

v. *Selection:* The new generation is determined by:

$$\boldsymbol{\theta}_{iG+1} = \begin{cases} \boldsymbol{\mu}_{iG} \text{ if } f(\boldsymbol{\theta}_{iG}) - f(\boldsymbol{\mu}_{iG}) > \varepsilon f(\boldsymbol{\theta}_{iG}), \\ \boldsymbol{\mu}_{iG} \text{ if } \left|f(\boldsymbol{\theta}_{iG}) - f(\boldsymbol{\mu}_{iG})\right| < \varepsilon f(\boldsymbol{\theta}_{iG}) \\ \qquad \text{and } \left\|\mathbf{A}^{\boldsymbol{\mu}_{i,G}}\right\| < \left\|\mathbf{A}^{\boldsymbol{\theta}_{i,G}}\right\|, \\ \boldsymbol{\theta}_{iG} \qquad \text{otherwise} \end{cases} \tag{10}$$

where $f(\cdot)$ is the fitness function and ϵ is a predefined tolerance rate. The DE process is repeated until the goal is met or a maximum learning epochs is reached. This algorithm can obtain a good generalization performance with compact networks. However, it does not obtain small norm of input weights and hidden layer biases. Note that a neural network can provide better generalization performance with small norm of network parameters.

3.2 Initialization of the first generation

In this section, we introduce another improvement of ELM, which is based on differential evolution and is called evolutionary least-squares extreme learning machine (ELS-ELM) (Hieu & Won, 2008). It utilizes a linear model for generating the initial population, and DE process for optimizing the parameters.

Let $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$ be the input matrix, and $\mathbf{W}=\begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_N \\ b_1 & b_2 & \ldots & b_N \end{bmatrix}$ be the matrix of input weights and biases. Then, the initialization of the first generation follows a linear model defined by:

$$\mathbf{XW=TP}, \tag{11}$$

where matrix $\mathbf{P} \in \mathbb{R}^{C \times N}$ should be assigned with random values.

The most commonly used method for the regularization of ill-posed problems has been Tikhonov regularization (Tikhonov & Arsenin, 1977), in which the solution for **W** of Eq. 11 can be replaced by seeking **W** that minimizes

$$\|\mathbf{WX}-\mathbf{TP}\|^2+\lambda\|\mathbf{W}\|^2, \tag{12}$$

where $\|\bullet\|$ is the Euclidean norm and λ is a positive constant. The solution for **W** is given by

$$\hat{\mathbf{W}}=(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{TP}. \tag{13}$$

In the case where **W** can be expressed as linear combination of $\mathbf{x}_j$ (j=1,2, …, n) and the number of features are large, we can obtain an indirect solution for Eq. 11 as:

$$\hat{\mathbf{W}}=\mathbf{X}^T\mathbf{Y} \tag{14}$$

where

$$\mathbf{Y}=(\mathbf{XX}^T+\lambda n\mathbf{I})^{-1}\mathbf{TP}. \tag{15}$$

Thus, ELS-ELM approach for training SLFNs can be described as follows:

1. **Initialization:** *Generate the population for the initial generation composed of parameter vectors* $\{\boldsymbol{\theta}_{i,G} \mid i=1, 2, \ldots, NP\}$, *where NP is the population size:*
   For each individual $\boldsymbol{\theta}_i$ in the population, we do:
   i. Randomly assign the values for the matrix **P**.
   ii. Estimate the input weights $\mathbf{w}_m$ and biases b_m of $\boldsymbol{\theta}_i$ by using Eq. 13 or 14.
   iii. Calculate the hidden layer output matrix **H** by using Eq. 5.
   iv. Determine the output weights **A** by using Eq. 8.
   v. Calculate the fitness value.
2. **Training process:**
   At each generation *G*, we do:
   i. *Mutation:* the mutant vector is generated by

   $$\mathbf{v}_{i,G+1}=\boldsymbol{\theta}_{r1,G}+F(\boldsymbol{\theta}_{r2,G}-\boldsymbol{\theta}_{r3,G}).$$

   ii. *Crossover:* the trial vector is formed using Eq. 9.
   iii. Compute the hidden layer output matrix **H** by Eq. 5.
   iv. Determine the output weights by Eq. 8.
   v. Evaluate the fitness for each individual.
   vi. *Selection:* The new generation is determined by Eq. 10.

This approach offers the improved performance for many applications, especially for regression problems.

4. Hematocrit estimation

Hematocrit is an important factor for medical procedures and hemodialysis, and is also the most highly affecting factor influencing the accuracy of glucose value measured by a hand-held device that uses the whole blood. Reports showed that the glucose measurement results are underestimated at higher hematocrit concentrations and overestimated at lower hematocrit levels (Kilpatrick et al., 1993; Louie et al., 2000; Tang et al., 2000). Consequently, estimating hematocrit concentrations plays an important role in enhancing performance of glucose measurements. Traditionally, this factor can be determined by centrifugation method performed in a small laboratory or by using automated analyzer. It can also be estimated by dielectric spectroscopy or some different techniques. However, most of the above approaches are quite complicated or require specific individual devices which are difficult to reduce the effects of hematocrit on glucose measurement by handheld devices.

With development of intelligent computional methods, simple methods for estimating hematocrit density while measuring glucose value with a biosensor-based handheld device should be investigated.

4.1 Methods

In this section, we present an approach based on DE optimation and neural networks for estimating hematocrit density. The SLFN architecture is proposed and trained by ELS-ELM algorithm. The input data are transduced current curves obtained during the process of glucose measurement by an electrochemical biosensor.

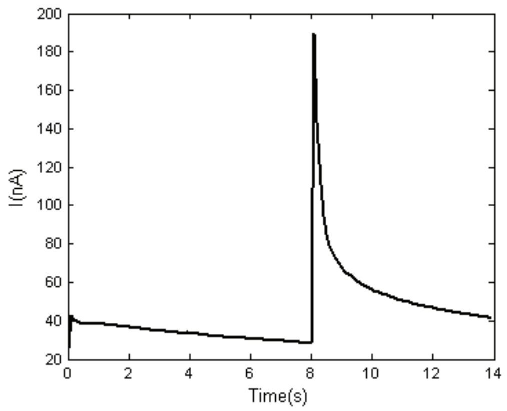

Fig. 2. Anodic Current Curve.

In the glucose measurement process by an electrochemical biosensor, the glucose oxidase(GOD) enzyme in biosensors is used to catalyze the oxidation of glucose by oxygen to produce gluconic acid and hydrogen peroxide, which is summarized as:

$$\text{Glucose}+O_2+\text{GO/FA}\rightarrow\text{Gluconic acid}+H_2O_2+\text{GO/FADH}_2$$
$$\text{GO/FADH}_2+\text{Ferricinium}^+\rightarrow\text{GO/FAD}+\text{Ferricinium}$$
$$\text{Ferrocence}\rightarrow\text{Ferrocence}^{+}+e^{-}$$

The reduced form of the enzyme (GO/FADH$_2$) is oxidized to its original state by an electron mediator (ferrocence). The resulting reduced mediator is then oxidized by the active electrode to produce the transduced anodic current. An example of the transduced anodic current curve obtained in the first 14 seconds is shown in Fig. 2. It was found that the data in the first eight seconds do not contain information for hematocrit estimation; it may be an incubation time while waiting for the enzyme reaction to be activated. Thus, in this study, we only focus on the second part of the current curve during the next six seconds. Note that this enzyme reaction characteristic may vary over biosensors from different manufacturers.
In the second period of six seconds, the anodic current curve is sampled at a frequency of 10Hz to produce the current points as shown in Fig. 3. There are 59 sampled current points used as the input features for networks. Denote $\mathbf{x}_j=[x_1, x_2, \ldots, x_{59}]^T$ as the j-th input pattern vector. Motivation of applying this anodic current curve $\mathbf{x}_j$ to SLFNs as the input vector for hematocrit estimation was introduced in (Hieu et al., 2008). It was also shown that the performance of hematocrit estimation can be improved with a new training algorithm which uses the DE process of ELS-ELM algorithm.

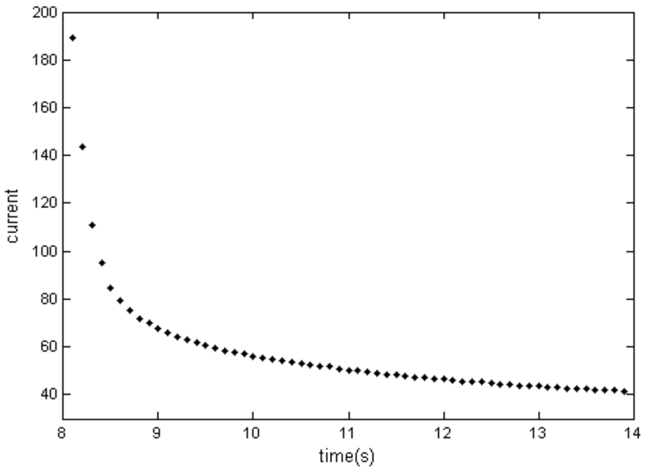

Fig. 3. The transduced current points used in hematocrit estimation.

4.2 Experimental results

In this section, performance of hematocrit estimation using SLFNs trained by ELM and DE process is presented. The data set of anodic current curves used in the experiments was obtained from 199 blood samples which were from randomly selected volunteers. For every blood sample, the accurate hematocrit density was first measured by the centrifugation method, and the anodic current curve was collected using a commercial handheld device for glucose measurement which uses an electrochemical biosensors. The distribution of reference hematocrit values collected from the blood samples used for this study by the centrifugation method is shown in Fig. 4 with mean 36.02 and deviation 6.39, which represents well the general distribution of hematocrit density values.
The data set was divided into training set (40%) and test set (60%). The input features were normalized into range [0, 1]. The algorithms were implemented on Matlab 7.0 environment.

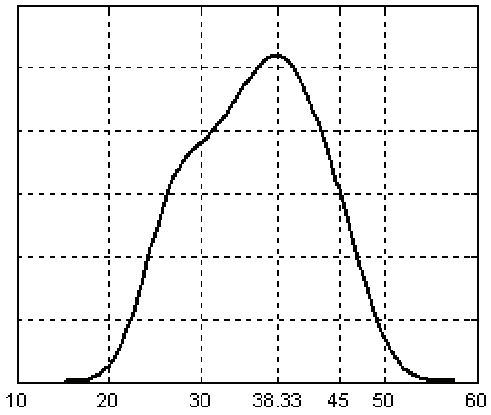

Fig. 4. Distribution of hematocrit density

Method	Training		Testing		# node
	RMSE	Mean	RMSE	Mean	
ELS-ELM	4.25	$\approx 10^{-5}$	4.63	-0.17	5
RLS-ELM	4.30	$\approx 10^{-7}$	4.91	-0.05	5
ELM	4.27	$\approx 10^{-4}$	4.90	-0.26	12

Table 1. Root mean square errors (RMSE) and the number of hidden units

The average results of fifty trials are shown in Table 1 with the root-mean-squared error(RMSE). In this table, the error was defined by the difference between the reference value measured by the centrifugation method and the output of the SLFN. From Table 1, we can see that the number of hidden units for ELS-ELM is equal to that for RLS-ELM (Hieu et al., 2008) while being much smaller than that for ELM. The errors for both training and testing are close to zero with the mean values of -0.17, -0.05 and -0.26 for ELS-ELM, RLS-ELM and ELM, respectively. The RMSE for ELS-ELM on test data set is 4.63, which empirically proves the outperformance of ELS-ELM compared to RLS-ELM (Hieu et al., 2008) and ELM. This improvement is significant to support for finding a method which can be used to reduce the effects of hematocrit density on glucose measurement by handheld devices.

5. Conclusion

Evolutionary algorithms (EAs) have been successfully applied in many complex optimization problems and engineering disciplines. Also, the single hidden-layer feedforward neural networks (SLFNs) have been widely used for machine learning due to their ability to form boundaries with arbitrary shape and to approximate any function with arbitrarily small error if the activation functions are chosen properly (Huang et al., 2000). Hematocrit density is an important factor for medical procedures and hemodialysis, and is the most highly affecting factor influencing the accuracy of glucose measurements with a hand-held device that uses the whole blood. Enzymatic reaction in glucose measurement with the electrochemical glucose biosensors is represented in the form of ion transfer to the electrode in the biosensor. Hematocrit density can be estimated while measuring the glucose value with the SLFN trained by combination of extreme learning machine (ELM) and differential evolution (DE) process.

This ionization of enzymatic reaction along time produces the anodic current curve, and is presented to SLFNs as the input vector. SLFNs trained by ELM, RLS-ELM and ELS-ELM are compared for hematocrit density estimation. It shows that they can be a good method for estimating and reducing the effect of hematocrit density on glucose measurement by handheld devices. Since this approach can provide the hematocrit density while measuring the glucose value, it can be simply implemented in the handheld devices to enhance the accuracy of glucose measurement which has to use the whole blood.

6. References

Hieu, T. H. & Won, Y. (2008). Evolutionary Algorithm for Training Single Hidden Layer Feedforward Neural Network, *The 2008 IEEE Int'l Joint Conference on Neural Networks (IJCNN2008)*, pp. 3027-3032.

Hieu, T. H.; Won, Y. & Kim, J. (2008). Neural Networks for the Estimation of Hematocrit from Transduced Current Curves, *Proc. of the 2008 IEEE Int'l Conf. on Networking, Sensing and Control*, pp.1517-1520.

Huang, G.-B.; Chen, Y.-Q. & Babri, H. A. (2000). Classification Ability of Single Hidden Layer Feedforward Neural Networks, *IEEE Trans. on Neural Networks*, Vol. 11, No. 3, pp. 799-801.

Huang, G.-B.; Zhu, Q.-Y. & Siew, C.-K. (2006). Extreme learning machine: Theory and applications, *Neurocomputing*, Vol. 70, pp. 489-501.

Ilonen, J.; Kamarainen, J. K. & Lampinen, J. (2003). Differential evolution training algorithm for feed forward neural networks, *Neural Processing Letters*, Vol. 17, pp. 145-163.

Kilpatrick, E. S.; Rumley, A. G. & Myin H. (1993). The effect of variations in hematocrit, mean cell volume and red blood count on reagent strip tests for glucose, *Ann Clin Biochem*, Vol. 30, pp. 485-487.

Louie, R. F.; Tang, Z., Sutton, D. V., Lee, J. H. & Kost, G. J. (2000). Point of Care Glucose Testing: effects of critical variables, influence of reference instruments, and a modular glucose meter design, *Arch Pathol Lab Med*, Vol. 124, pp. 257-266.

Masters, T. & Land, W. (1997). A new training algorithm for the general regression neural network, *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, pp. 1990-1994.

Serre, D. (2002). *Matrices: Theory and Applications*, Springer, New York.

Storn, R. & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341-359.

Subudhi, B. & Jena, D. (2008). Differential Evolution and Levenberg Marquardt Trained Neural Network Scheme for Nonlinear System Identification, *Neural Processing Letters*, Vol. 27, pp. 285-296.

Tang, Z.; Lee, J. H., Louie, R. F., Kost, G. J. & Sutton, D. V. (2000). Effects of Different Hematocrit Levels on Glucose Measurements with HandHeld Meters for Point of Care Testing", *Arch Pathol Lab Med*, Vol. 124, pp. 1135-1140.

Tikhonov, A. N. & Arsenin, A. V. (1977). *Solution of Ill-posed Problems*, Winston & Son, Washingston.

Zhu, Q.-Y.; Qin A.K., Suganthan P.N. & Huang G.-B. (2005). Evolutionary Extreme Learning Machine, *Pattern Recognition*, Vol. 38, pp. 1759-1763.

Evolutionary Computation
Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7
Hard cover, 572 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.