
Neural Processing Letters **17:** 93–105, 2003.

Differential Evolution Training Algorithm for Feed-Forward Neural Networks

JARMO ILONEN, JONI-KRISTIAN KAMARAINEN and JOUNI LAMPINEN
Laboratory of Information Processing, Lappeenranta University of Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland. e-mail: {ilonen;jkamarai}@lut.fi

Abstract. An evolutionary optimization method over continuous search spaces, differential evolution, has recently been successfully applied to real world and artificial optimization problems and proposed also for neural network training. However, differential evolution has not been comprehensively studied in the context of training neural network weights, i.e., how useful is differential evolution in finding the global optimum for expense of convergence speed. In this study, differential evolution has been analyzed as a candidate global optimization method for feed-forward neural networks. In comparison to gradient based methods, differential evolution seems not to provide any distinct advantage in terms of learning rate or solution quality. Differential evolution can rather be used in validation of reached optima and in the development of regularization terms and non-conventional transfer functions that do not necessarily provide gradient information.

Key words. differential evolution, evolutionary algorithms, feed-forward neural network, neural network training

1. Introduction

Since the development of the back-propagation method, many modified and new algorithms have been proposed for training feed-forward neural networks; many of them having a very fast convergence rate for reasonable size networks. However, the number of suitable network training algorithms dramatically decreases when the neural network training becomes a large scale, i.e., the number of network parameters grows drastically. For example, learning a huge number of hidden layer weights in a multi-layer perceptron (MLP) neural network can be considered as a large scale optimization problem. On the other hand, global optimization methods are under continuous development, and lately, genetic algorithms and evolution strategies have been studied and found to be promising stochastic optimization methods [1]. The perceived advantages of genetic algorithms and evolution strategies as optimization methods motivated the authors to consider such stochastic methods in the context of training artificial neural networks and optimizing the structure of the networks. A survey and overview of evolutionary algorithms in evolving artificial neural networks can be found in [24].

Differential evolution (DE), first introduced in [20], is successfully applied to many artificial and real optimization problems and applications [9], such as aerodynamic

shape optimization [18], automated mirror design [4], optimization of radial active magnetic bearings [22], optimization of fermentation using a high ethanol tolerance yeast [23], and mechanical engineering design [10]. A differential evolution based neural network training algorithm, first introduced in [13], was proposed in the form used in this study by one of the authors in [25], where the method's characteristics as a global optimizer were compared to other neural network training methods. Similar results were also reported by others in [5, 6, 19]. More comprehensive studies are needed to reliably evaluate the necessity of using differential evolution or other stochastic methods in the training of artificial neural networks, and to establish the kind of advantages contemporary stochastic methods can provide.

To prove one training algorithm to be superior to others, even in some limited domain, is not a simple task due to the difficulties of measuring and comparing the performance of different types of methods. Since the learning algorithm can be viewed as a non-linear optimization algorithm, all aspects, such as effectiveness, efficiency, and robustness, should be considered when evaluating alternative methods. Generally, the comparison can be made by using as a performance measure the computation time of training needed to reach a given error rate. The use of an epoch as a measure can be misleading because of the difficulties in defining the epoch for both evolutionary methods and gradient-methods. In particular, the computational cost for the epoch may vary significantly from one algorithm to another. Other important issues in testing and benchmarking training algorithms are briefly described in [17] and studied in more detail, including the data sets to use, in [16]. In the current work the authors study whether differential evolution can replace or even be compared to back-propagation methods in training feed-forward MLP neural networks. The comparison is based on measuring training performance of different type of algorithms.

Training MLPs can be considered as a difficult global optimization problem, despite the fact that local optimizers are usually applied for training. Investigation of applying global optimizers to training is well-motivated, since local optimizers have basically limited capabilities for global optimization. A further motivation comes from the need to apply transfer function or regularization approaches that do not satisfy the requirements concerning the availability of gradient information. Convergence to a locally optimal solution is a fundamental limitation of any local search based training approach.

2. Training Algorithms for Neural Networks

Any non-linear optimization method, a local or global one, can be applied to the optimization of feed-forward neural networks weights. Naturally, local searches are fundamentally limited to local solutions, while global ones attempt to avoid this limitation. The training performance varies depending on the objective function and underlying error surface for a given problem and network configuration. Since the gradient information of error surface is available for the most widely

applied network configurations, the most popular optimization methods have been variants of gradient based back-propagation algorithms. Of course, this is sometimes the result of an inseparable combination of network configuration and training algorithm which limits the freedom to choose the optimization method. Widely applied methods are, for example, modified back-propagation [11], back-propagation using the conjugate-gradient approach [2], scaled conjugate-gradient and its stochastic counterpart [14], the Marquadt algorithm [7], and a concept learning based back-propagation [8]. Many of these gradient based methods are studied and discussed even for large networks in [14]. Several methods have been proposed for network configurations where the gradient information is not available, such as simulated annealing for networks with non-differentiable transfer functions [3].

In many studies only small network configurations are considered in training experiments. Many gradient based methods and especially the Levenberg-Marquadt method are extremely fast for small networks (few hundreds of parameters), thus, leaving no room or motivation for discussion of using evolutionary approaches in the cases where the required gradient information is available. The problem of local minima can be efficiently avoided for small networks by using repeated trainings and randomly initialized weight values. Nevertheless, evolutionary based global optimization algorithms may be useful for validation of an optimal solution achieved by a gradient based method. The problem of local minima and successfully applied differential evolution based training are both discussed in [25] and evolutionary methods evolving neural networks (architecture, learning rules, and connection weights) in general in [24]. In addition to the problem with local minima, generalization and over-fitting are also important considerations.

For large feed-forward neural networks, consisting of thousands of neurons, the most efficient training methods (Levenberg-Marquadt, Quasi-Newton, etc.) demand an unreasonable amount of computation due to their computational complexity in time and space. Basic gradient-based methods and stochastic approaches can however often be used, but the problem of their low training convergence rate must be considered. One possibility could be a hybrid of traditional optimization methods and evolutionary algorithms as studied in [12, 13, 15]. Unfortunately, it seems that none of the contemporary methods can offer superior performance over all other methods on all problem domains. It seems that no single solution appears to be available for the training of artificial neural networks.

3. Differential Evolution Training Algorithm

Differential evolution can be classified as a floating-point encoded evolutionary algorithm for global optimization over continuous spaces. As such, it can be applied to global searches within the weight space of a typical feed-forward neural network.

Output of a feed-forward neural network is a function of synaptic weights $\mathbf{W}$ and input values $\mathbf{x}$, i.e., $\mathbf{y} = f(\mathbf{x}, \mathbf{W})$. In standard training processes, both the input vector

$\mathbf{x}$ and the output vector $\mathbf{y}$ are known, and the synaptic weights in $\mathbf{W}$ are adapted to obtain appropriate functional mappings from the input $\mathbf{x}$ to the output $\mathbf{y}$. Generally, the adaptation can be carried out by minimizing the network error function E which is of the form

$$E(\mathbf{y}, f(\mathbf{x}, \mathbf{W})) : (\mathbf{y}^{D_1}, \mathbf{x}^{D_2}, \mathbf{W}^{D_3}, f) \to R. \tag{1}$$

The optimization goal is to minimize the objective function $E(\mathbf{y}, f(\mathbf{x}, \mathbf{W}))$ by optimizing the values of the network weights (now $D = D_3$)

$$\mathbf{W} = (w_1, \ldots, w_D). \tag{2}$$

Similar to other evolutionary algorithms, DE operates on a population, $\mathbf{P}_G$, of candidate solutions, not just a single solution. These candidate solutions are the individuals of the population. DE maintains a population of constant size that consists of NP, real-value vectors, $\mathbf{W}_{i,G}$, where i is index to the population and G is the generation to which the population belongs.

$$\mathbf{P}_G = (\mathbf{W}_{1,G}, \ldots, \mathbf{W}_{NP,G}), \quad G = 0, \ldots, G_{\max} \tag{3}$$

Additionally, in network training each vector contains D network weights (chromosomes of individuals):

$$\mathbf{W}_{i,G} = (w_{1,i,G}, \ldots, w_{D,i,G}), \quad i = 1, \ldots, NP, \quad G = 0, \ldots, G_{\max} \tag{4}$$

DE's self-referential population reproduction scheme is different from other evolutionary algorithms. After initialization of the first population, vectors in the current population, $\mathbf{P}_{G+1}$, are randomly sampled and combined to create candidate vectors for the subsequent generation, $\mathbf{P}_{G+1}$. The population of candidates, trial vectors $\mathbf{P}'_{G+1} = \mathbf{U}_{i,G+1} = \mathbf{u}_{j,i,G+1}$ is generated as follows

$$\begin{aligned} v_{j,i,G+1} &= w_{j,r3,G} + F \cdot (w_{j,r1,G} - w_{j,r2,G}) \\ u_{j,i,G+1} &= \begin{cases} v_{j,i,G+1}, & \text{if} \quad \text{rand}_j[0, 1) \leqslant CR \\ w_{j,i,G}, & \text{otherwise,} \end{cases} \end{aligned} \tag{5}$$

where

$$\begin{aligned} &i = 1, \ldots, NP, \quad j = 1, \ldots, D \\ &r_1, r_2, r_3 \in \{1, \ldots, NP\}, \text{ randomly selected, except } r_1 \neq r_2 \neq r_3 \neq i \\ &CR \in [0, 1], \quad F \in (0, 1+]. \end{aligned}$$

Now, CR is a real-valued crossover factor that controls the probability that a trial vector parameters will come from the randomly chosen, mutated vector $v_{j,i,G+1}$ instead of the current vector $w_{j,i,G}$. In general, F and CR affect the convergence speed and robustness of the search process. Their optimal values depend both on objective function characteristics and the population size NP, and thus, the selection of optimal parameter values is an application dependent task [20, 21].

DE's selection scheme also differs from other evolutionary algorithms. The population for the next generation, $\mathbf{P}_{G+1}$, is selected from the current population $\mathbf{P}_G$ or the child population according to the following rule

$$\mathbf{W}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G+1}, & \text{if } E(\mathbf{y}, f(\mathbf{x}, \mathbf{W}_{i,G+1}) \leqslant E(\mathbf{y}, f(\mathbf{x}, \mathbf{W}_{i,G})) \\ \mathbf{W}_{i,G}, & \text{otherwise} \end{cases}$$

Thus, each individual of the temporary population is compared to its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. The interesting point concerning DE's replacement scheme is that a trial vector is only compared to one individual, not to all individuals in the current population. It is also noteworthy that the replacement scheme ensures that the population does not diverge from or lose the best solution found so far.

4. Experiments

In this section the performance of differential evolution in feed-forward neural network training is evaluated by experiment. The experiments were carried out using various configurations of multi-layer perceptron (MLP) neural network and training problems from two commonly used problem domains: pattern recognition and function approximation. A Matlab implementation of the differential evolution training algorithm used in the experiments is available at *http://www.it.lut.fi/project/nngenetic/*, and the data sets used, recognized by the data set name, are downloadable from public data repositories (links available at *http://www.it.lut.fi/project/nngenetic/*).

4.1. EFFICIENCY

An efficient algorithm ensures that the solution can be found in reasonable time. The efficiency can be assessed by estimating the computational complexity (time and space) of a method. In this experiment, the computing time of one epoch was recorded for various MLP training algorithms. Training set data consisted of 1000 sample points from the $\sin(x)$ function and a network structure of $1 - n - 1$ was used. 6 different training algorithms were applied to the training of the networks with a different number n of hidden layer units.

As shown in Figure 1, the gradient methods that converge fast for a small number of hidden units (Levenberg-Marquadt, quasi-Newton) and DE with population size linearly dependent on the number of weights need an intolerable amount of computation and thus are unsuitable for training large MLP networks. To achieve even poor results efficiently only the basic gradient methods (gradient descent, scaled conjugate gradient), simple optimization methods (one-step secant), and the differential evolution with constant population size can be applied.

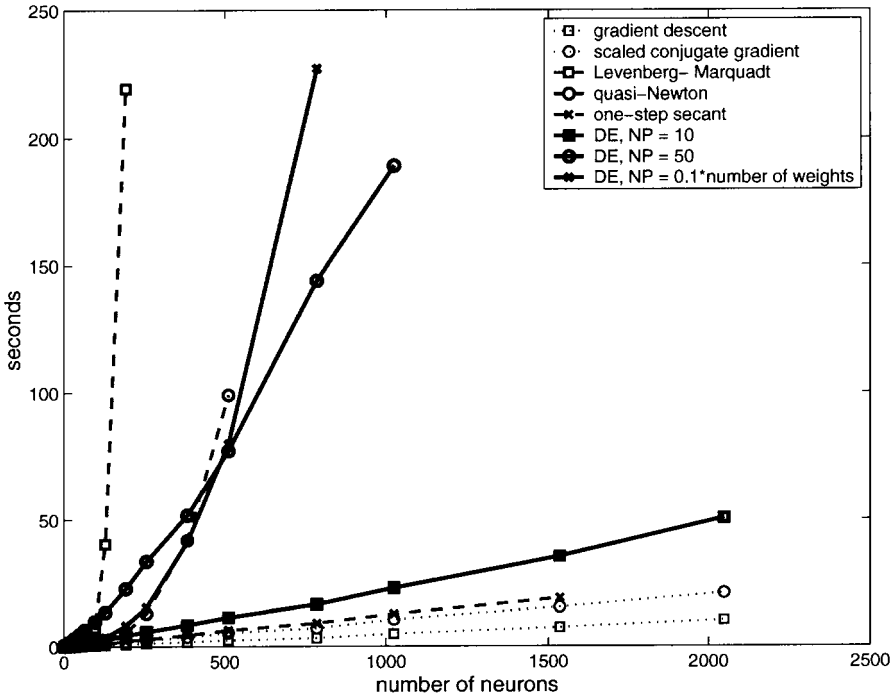


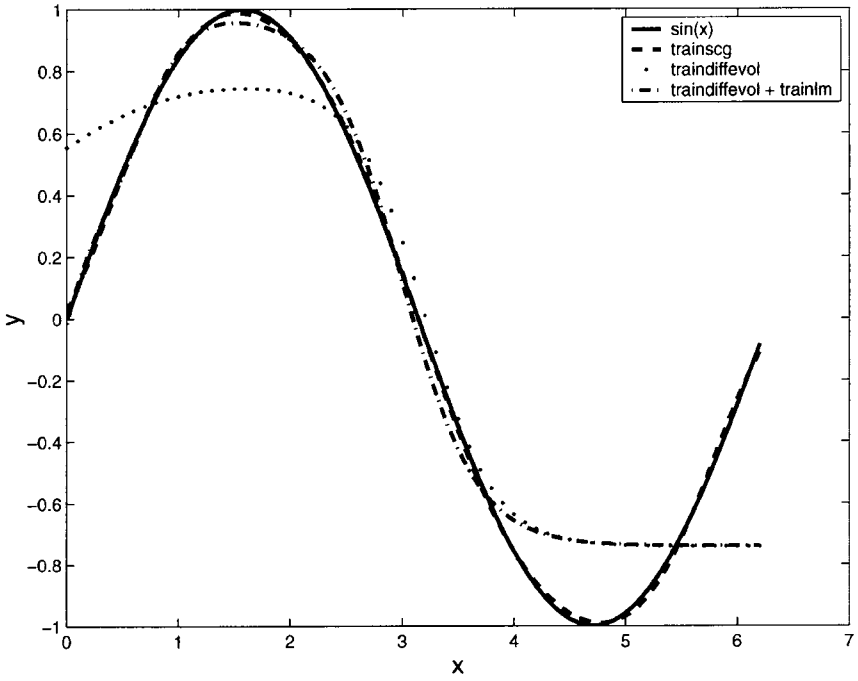
Figure 1. Computing times of one epoch for training algorithms.

The interpretation of the results is however not straightforward. If the computation to reach a given error level (effectiveness aspect) takes an exponential number of training epochs, then the advantage of the faster computation time (linear) for a single epoch may be lost for large scale training problems. Also, the definition of an epoch is vague and may differ for different methods, and thus, epochs cannot be unambiguously compared (concerns also results in Figure 1).

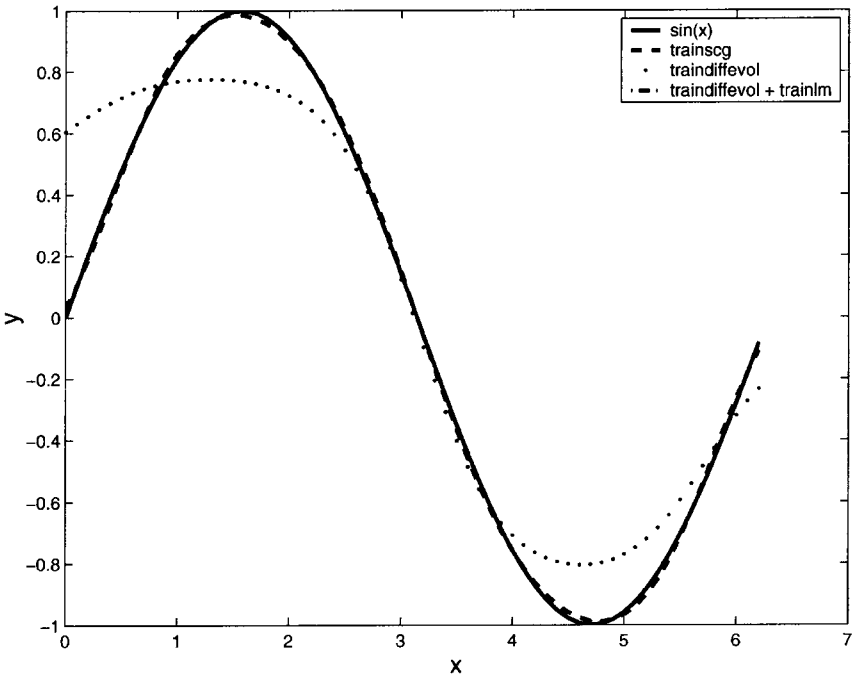
4.2. ROBUSTNESS

A robust method finds the optimum in most cases, not only random. For example, gradient based methods are often very robust if proper initialization of the network weights is used (e.g. Nguyen-Widrow). The robustness of differential evolution may be estimated by repeating training and evaluating the type of optimums (local/global). In this experiment, 1000 sample points of $\sin(x)$ function for $x = [0, 2\pi]$ and a MLP structure $1 - logsig(3) - purelin(1)$, where $logsig(n)$ means n neurons with a sigmoid transfer function and $purelin(n)$ n neurons with a pure linear transfer function, were used. The training time was limited to 5 seconds and the population size NP was the same as the number of weights D.

It is evident that the differential evolution parameters CR and F affect not only the convergence rate but also the robustness, as demonstrated in Figure 2. Figure 2 shows



$CR = 0.5,\ F = 0.8$



$CR = 0.3,\ F = 0.4$

Figure 2. Results of training experiments with sin(x) data.

the original training data (sin(x)) and network output after training with the scaled conjugate gradient (trainscg), differential evolution (traindiffevol), and Levenberq-Marquadt applied to the optimum found by differential evolution (traindiffevol + trainlm). The Levenberq-Marquadt method was used to prove the type of the solution found by DE (local or global). It should be noted that due to the stochastic behavior of the DE, the solutions vary from one run to another, and thus, the plots in Figure 2 represent only one possible candidate. However, the DE method can be considered as a robust method, that is, it finds the optimum in most cases.

Because the MLP neural network may have an infinite number of optimal solutions of the same performance value for the given training set, the network weight values may increase explosively for the differential evolution training algorithm. In experiments, it was found that if a weight penalty term was added to the error function, the differential evolution seemed to converge to the same values as the gradient methods. It seems that the solution of the differential evolution training algorithm is more stable if some kind of generalization factor, for example, a weights penalty term is included.

4.3. EFFECTIVENESS

The effectiveness of differential evolution was estimated by measuring the mean squared error and mean squared weights and biases (MSEREG) for a training period of restricted time. In the experiment, the differential evolution training with several different population sizes NP was compared to other commonly used methods. Benchmark problems and the network configurations of this experiment are shown in Table I. The network weights were initialized by the Nguyen-Widrow initialization method to provide good initial weights for the gradient based methods and a penalty ratio 0.1 was used for weights and biases. In three problems, SIN, CHOLESTEROL and DIABETES, the results of DE are comparable to other methods (min MSEREG), but for the CANCER problem DE had a slower convergence rate than the others, as shown in Table II and in Figures 3 and 4.

5. Discussion and Conclusions

In this study, a differential evolution (DE) training algorithm was used to train feedforward multilayer perceptron neural networks. In the experiments where a penalty

Table I. Benchmark problems used.

Problem title	Problem type	Network structure	Training time
SIN	Function Approx.	1-5-1	60
CANCER	Pattern Recog.	9-5-5-2	180
CHOLESTEROL	Function Approx.	21-15-3	600
DIABETES	Pattern Recog.	8-15-15-2	900

Table II. Training results over several independent trials.

Problem	Method	Mean MSEREG	Min MSEREG	Max MSEREG
SIN	traingdx	0.1044	0.1040	0.1069
	trainoss	0.1045	0.1040	0.1047
	trainrp	–	–	–
	trainscg	0.1043	0.1040	0.10472
	trainde NP = 20	0.1285	0.1051	0.3265
	trainde NP = 250	0.1285	0.1489	0.3278
CANCER	traingdx	0.0281	0.0277	0.0294
	trainoss	0.0277	0.0277	0.0280
	trainrp	–	–	–
	trainscg	0.0277	0.0277	0.0277
	trainde NP = 50	0.1640	0.1134	0.2267
	trainde NP = 250	0.2127	0.1963	0.2276
CHOLESTEROL	traingdx	774.50	515.60	1329.4
	trainoss	656.90	343.70	1249.4
	trainrp	209.01	204.10	221.00
	trainscg	108.66	91.567	134.11
	trainde NP = 50	896.80	749.20	1184.4
	trainde NP = 250	1168.8	941.00	1344.2
DIABETES	traingdx	0.1327	0.1299	0.1359
	trainoss	0.1092	0.0997	0.1211
	trainrp	0.1277	0.1218	0.1324
	trainscg	0.1012	0.0990	0.1029
	trainde NP = 50	0.2226	0.2118	0.2332
	trainde NP = 250	0.2478	0.2346	0.2597

term for network weights and biases (MSEREG) was included, the performance of the differential evolution training algorithm appeared to be comparable to that of the gradient based methods. Due to its scalability, it is evident that the DE method can be applied to the training of enormous neural networks. However, our current study did not reveal any distinct advantage over gradient based methods, and thus, no evidence is found in this study to prefer differential evolution over gradient methods. It seems that if the gradient information is available, it should be used in the optimization. Correspondingly, if the error surface is very rough and the gradient information frequently leads to local optimums, stochastic approaches and global optimization methods in general may be the only practical alternatives. Differential evolution may however be more useful in the special case of error surfaces with characteristics from both smooth and rough surfaces, yet even in this case, some kind of hybrid algorithm utilizing both evolutionary optimization and gradient information could be the most beneficial.

Currently, the superior properties of differential evolution are in areas other than the training performance. The main advantages of differential evolution are: (1) no major restrictions apply to the error function, e.g., non-differentiable transfer

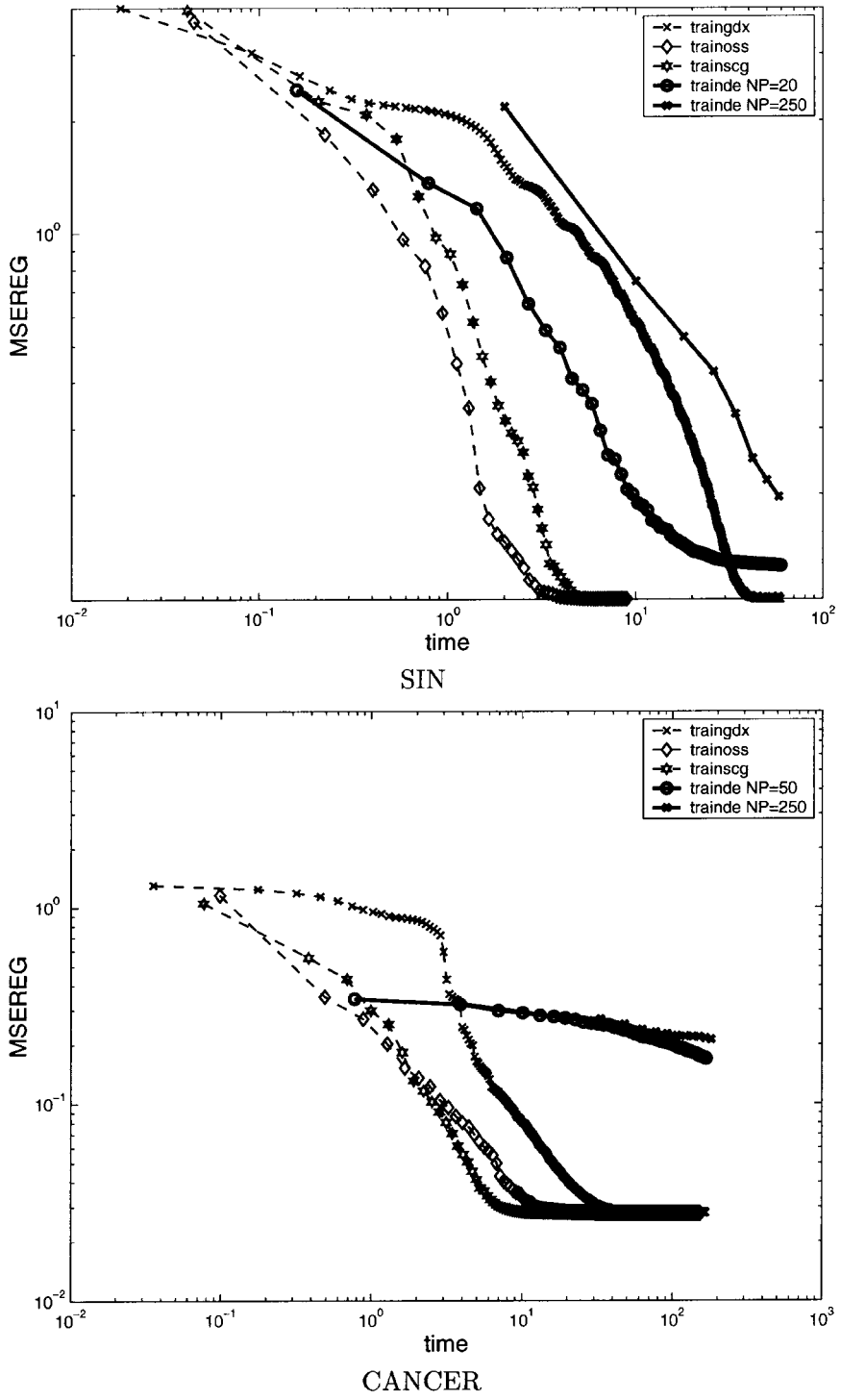
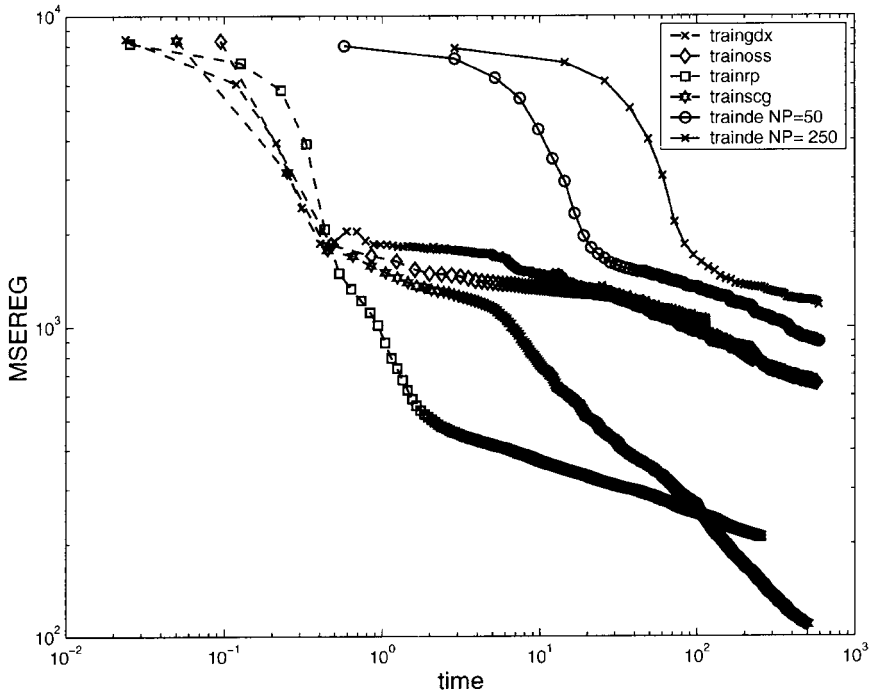
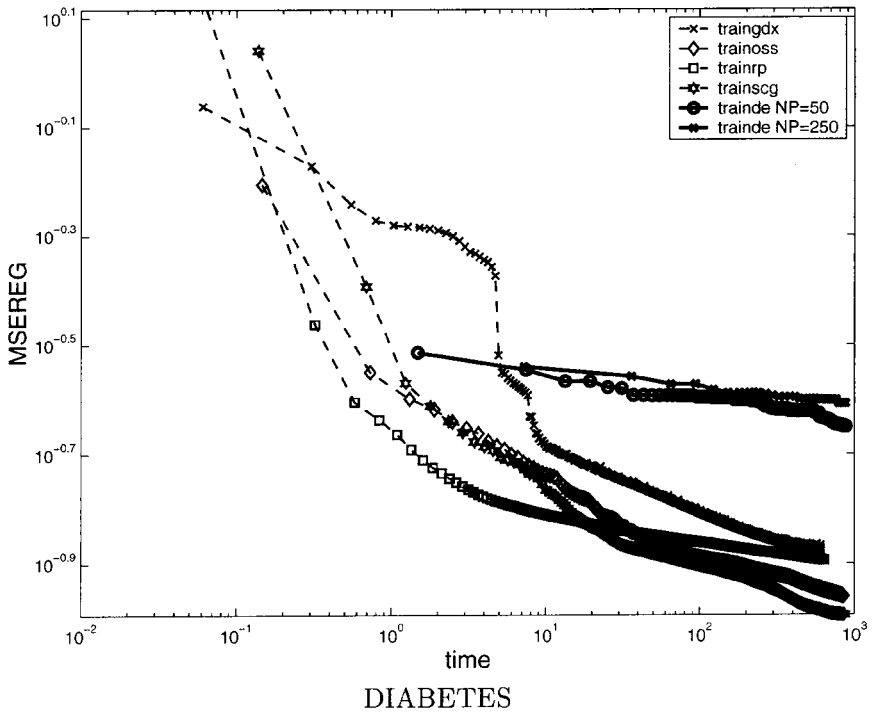


Figure 3. Results for benchmark problems SIN and CANCER.



CHOLESTEROL



DIABETES

Figure 4. Results for benchmark problems CHOLESTEROL and DIABETES.

functions may be used; (2) there are no major restrictions on the regularization methods; (3) convergence to a global minimum can be expected (but the time needed for convergence can be intolerable); (4) easy tuning of the algorithm parameters (mainly the size of population); (5) the linear time and space complexity of the algorithm can be established.

The most interesting properties are properties (1) and (2). Due to the nature of differential evolution, there are no special restrictions on the performance or regularization functions, and thus, new approaches to regularization and performance can be studied. Property (3) permits the use of differential evolution in validation of networks trained with gradient methods, i.e., the optimum found by the DE is never worse than the initial optimum found by a gradient based method, but it is highly probable that the DE converges to a better optimum if such exists.

It seems that differential evolution algorithm provides new interesting topics for the training of feed-forward neural networks, such as, why some local optimums are more attractive to differential evolution and why differential evolution finds a global minimum for some problem configurations when gradient-based methods do not, and finally what is the effect of a selected performance function on the optimum found by DE? These problems will be addressed in future work.

Acknowledgements

The authors would like to thank ECSE (East Finland Universities Graduate School in Computer Science and Engineering), and the Emil Aaltonen and TES (Tekniikan edistämissäätiö) foundations for financial support.

References

1. Alander, J. T.: An Indexed Bibliography of Genetic Algorithms and Neural Networks, ftp.uwasa.fi/cs/report94-1/gaNNbib.ps.Z., 1999.
2. Charalambous, C.: Conjugate gradient algorithm for efficient training of artificial neural networks, *IEE G* (*Circuits, Devices and Systems*), **139**(3) (1992), 301–310.
3. Day, S. P. and Camporese, D. S.: A stochastic training technique for feed-forward neural networks, In: *IJCNN International Joint Conference on Neural Networks*, **3** (1990), 607–612.
4. Doyle, S., Corcoran, D. and Connell, J.: Automated mirror design using an evolution strategy, *Optical Engineering*, **38**(2) (1999), 323–333.
5. Fischer, M. M., Hlavackova-Schindler, K. and Reismann, M.: A global search procedure for parameter estimation in neural spatial interaction modelling, *Regional Science*, **78**(2) (1999), 119–134.
6. Gang, L., Yiqing, T. and Fu, T.: A fast evolutionary algorithm for neural network training using differential evolution, In: *ICYCS'99 Fifth International Conference for Young Computer Scientists*, **1** (1999), 507–511.
7. Hagan, M. T. and Menhaj, M. B.: Training feedforward networks with the Marquadt algorithm, *IEEE Transactions on Neural Networks*, **5**(6) 1994.
8. Japkowicz, N. and Hanson, S. J.: Adaptability of the backpropagation procedure, In: *IJCNN'99 International Joint Conference on Neural Networks*, **3** (1999), 1710–1715.

9. Lampinen, J.: A Bibliography of Differential Evolution Algorithm, http: //www.lut.fi/~jlampine/debiblio.htm., 2001.
10. Lampinen, J. and Zelinka, I.: *New Ideas in Optimization*, Chapt. Mechanical Engineering Design Optimization by Differential Evolution, McGraw-Hill, (1999), pp. 127–146.
11. Liang, M. Wang, S.-X. and Luo Y.-H.: Fast learning algorithms for multi-layered feedforward neural network, In: *IEEE 1994 National Aerospace and Electronics Conference NAECON 1994*, **2** (1994), 787–790.
12. Magoulas, G., Plagianakos, V. and Vrahatis, M.: Hybrid methods using evolutionary algorithms for on-line training, In: *Proceedings of IJCNN'01, International Joint Conference on Neural Networks*, **3** (2001), 2218–2223.
13. Masters, T. and Land, W.: A new training algorithm for the general regression neural network, In: *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, **3** (1997), 1990–1994.
14. Moller, M.: Efficient training of feed-forward neural networks, Ph.D. thesis, Computer Science Department, Aarhus University, Arhus, Denmark, 1997.
15. Neelaveni, R., Gurusamy, G. and Hemavathy, L.: Adaptive genetic algorithm and differential evolution based backpropagation neural network for epileptic pattern recognition, In: *Proceedings of the National Conference on Technology Convergence for Information, Communication and Entertainment*, (2001), 26–30.
16. Prechelt, L.: PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms, Technical report, Fakultat fur Informatik, Universitat Karlsruhe, Germany, 1994.
17. Prechelt, L.: Some notes on neural learning algorithm benchmarking, *Neurocomputing*, **9**(3) (1995), 343–347.
18. Rogalsky, T., Kocabiyik, S. and Derksen, R.: Differential evolution in aerodynamic optimization, *Canadian Aeronautics and Space Journal*, **46**(4) (2000), 183–190.
19. Schmitz, G. P. and Aldrich, C.: Combinatorial Evolution of Regression Nodes in Feedforward Neural Networks, *Neural Networks*, **12**(1) (1999), 175–189.
20. Storn, R. and Price, K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, USA (http://www.icsi.berkeley.edu/techreports/1995.abstracts/tr-95-012.html), 1995.
21. Storn, R. and Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, **11**(4) 1997, 341–359.
22. Stumberger, G., Dolinar, D., Pahner, U. and Hameyer, K.: Optimization of radial active magnetic bearings using the finite element technique and differential evolution algorithm, *IEEE Transactions on Magnetics*, **36**(4) (2000), 1009–1013.
23. Wang, F.-S. and Sheu, J.-W.: Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast, *Chemical Engineering Science*, **55**(18) (2000), 3685–3695.
24. Yao, X.: Evolving artificial neural networks, *Proceedings of the IEEE*, **87**(9) (1999), 1423–1447.
25. Zelinka, I. and Lampinen, J.: An evolutionary learning algorithms for neural networks, In: *5th International Conference on Soft Computing MENDEL'99*, (1999), 410–414.