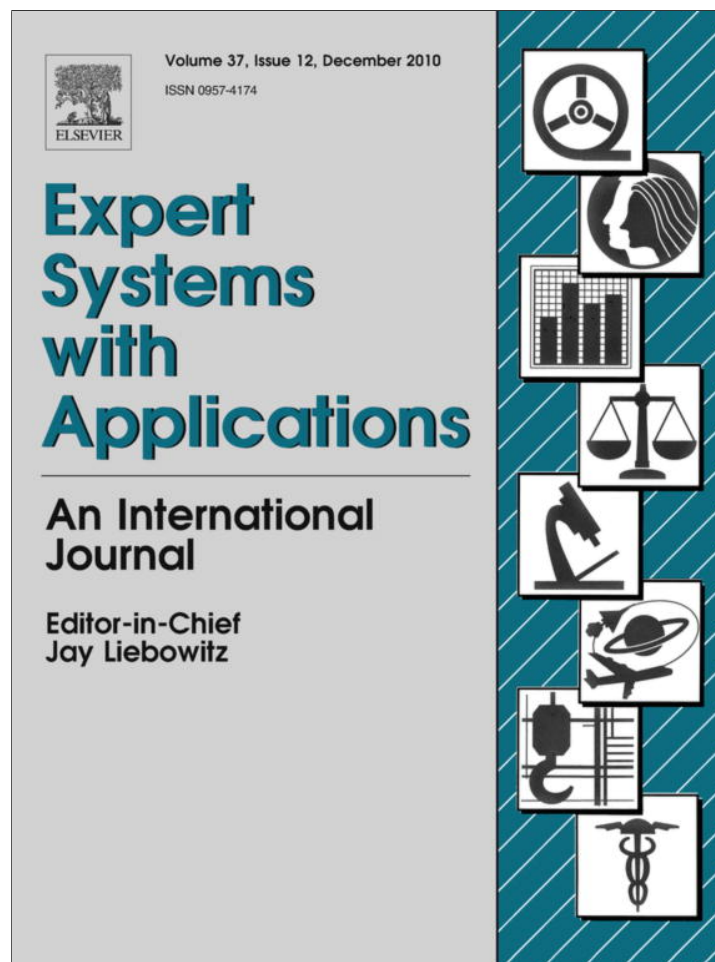


Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Contents lists available at ScienceDirect

Expert Systems with Applications

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

Evaluation of global and local training techniques over feed-forward neural network architecture spaces for computer-aided medical diagnosis

Turker Ince^{a,*}, Serkan Kiranyaz^b, Jenni Pulkkinen^b, Moncef Gabbouj^b

^aIzmir University of Economics, Faculty of Engineering and Computer Science, Izmir, Turkey

^bTampere University of Technology, Tampere, Finland

ARTICLE INFO

Keywords:

Artificial neural networks

Backpropagation

Particle swarm optimization

ABSTRACT

In this paper, we investigate the performance of global vs. local techniques applied to the training of neural network classifiers for solving medical diagnosis problems. The presented methodology of the investigation involves systematic and exhaustive evaluation of the classifier performance over a neural network architecture space and with respect to training depth for a particular problem. In this study, the architecture space is defined over feed-forward, fully-connected artificial neural networks (ANNs) which have been widely used in computer-aided decision support systems in medical domain, and for which two popular neural network training methods are explored: conventional backpropagation (BP) and particle swarm optimization (PSO). Both training techniques are compared in terms of classification performance over three medical diagnosis problems (*breast cancer*, *heart disease*, and *diabetes*) from *Proben1* benchmark dataset and computational and architectural analysis are performed for an extensive assessment. The results clearly demonstrate that it is not possible to compare and evaluate the performance of the two algorithms over a single network and with a fixed set of training parameters, as most of the earlier work in this field has been carried out, since training and test classification performances vary significantly and depend directly on the network architecture, the training depth and method used and the available dataset. We, therefore, show that an extensive evaluation method such as the one proposed in this paper is basically needed to obtain a reliable and detailed performance assessment, in that, we can conclude that the PSO algorithm has usually a better generalization ability across the architecture space whereas BP can occasionally provide better training and/or test classification performance for some network configurations. Furthermore, we can in general say that the PSO, as a global training algorithm, is capable of achieving minimum test classification errors regardless of the training depth, i.e. shallow or deep, and its average classification performance shows less variations with respect to network architecture. In terms of computational complexity, BP is in general superior to PSO for the entire architecture space used.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Artificial neural networks (ANNs) are known as “universal approximators” and “computational models” with particular characteristics such as the ability to learn or adapt, to organize or to generalize data. Based on the literature in medical domain, computer-aided diagnostic systems with embedded artificial intelligence algorithms have increasingly been used to assist medical experts in diagnosing a patient (Lisboa & Taktak, 2006). Due to their automatic (self-adaptive) process and capability to learn complex, nonlinear surfaces among different classes, ANN classifiers have become a popular choice for decision support in medical diagnosis and have been shown to be effective in the clinical do-

main (Lisboa, 2002). Most of these neural network classifier studies have used feed-forward, fully-connected ANNs, the well-known multilayer perceptrons (MLPs), with the back-propagation (BP) training algorithm (Lisboa & Taktak, 2006). BP is a gradient descent method on an error surface, which has a local search capability. However, such a capability causes it to get trapped into the nearest local minimum, and thus the training outcome becomes entirely dependent on the initial (weight) settings (Kolen & Pollack, 1990). Therefore, a single BP run is in general unreliable, unrepeatable and sub-optimum, especially in the case of a large number of local minima. In practice, the performance of a BP-ANN classifier depends on the particular pattern recognition or classification problem, description of the input space (extracted features), training sample size, and initial settings of the parameters (i.e. weights, learning rate, and momentum). Currently, there are many variants and extensions of BP to address some of these issues, which in-

* Corresponding author. Tel.: +90 2324888509; fax: +90 2324888475.
E-mail address: turker.ince@ieu.edu.tr (T. Ince).

clude gradient descent with momentum, scaled conjugate gradient (SCG), resilient propagation (RPROP), BFGS quasi-Newton, and Levenberg–Marquardt (LM) algorithms (Gudise & Venayagamoorthy, 2003). However, the practitioner must still choose the correct parameter settings with the corresponding algorithm for a specific network and a particular problem.

For many real-world applications, which can be formulated as complex, nonlinear, and multi-modal problems, the global search techniques may perform better since they are capable of finding the global optimum solutions; however, this is never guaranteed. As a recent optimization technique, the particle swarm optimization (PSO) was proposed by Kennedy and Eberhart (1995). It is a population based stochastic search and optimization process. PSO originated from computer simulation of the individuals (particles or living organisms) in a bird flock or a fish school which basically show a natural behavior when they search for some target (e.g. food). The goal is, therefore, to converge to the global optimum of some multi-dimensional and possibly nonlinear function or system. In principle, PSO follows the same path as the existing evolutionary algorithms (EAs). In addition to strong global searching ability, the EA family of algorithms have the advantage of being applicable to any type of ANN, feed-forward or not, with any activation function, differentiable or not. Several researchers have successfully applied PSO for training feed-forward (Carvalho & Ludermir, 2007; Meissner, Schmuken, & Schneider, 2006; Yu, Xi, & Wang, 2007) and recurrent ANNs (Hu & Shi, 2004; Settles, Rodebaugh, & Soule, 2003) to solve classification problems.

Many studies for performance evaluation of neural network classifiers using local and global learning algorithms have recently been presented in the literature. In Hosseini, Luo, and Reynolds (2006), the performance of six feed-forward ANN architectures (four single-hidden layer and two double-hidden layer networks) is investigated for electrocardiogram (ECG) signal diagnosis. The double-hidden layer ANN classifier is shown to enhance the ECG signal classification process. In another study (Van den Bergh, 2002), the author compared the training and testing (generalization) performance of PSO-based algorithms in terms of mean square error (MSE) with BP and Genetic Algorithm (GA)-based techniques. In this study, simple one-hidden layer MLPs with a suitable number of hidden units that are determined by an ANN pruning technique for each problem were used to solve a variety of classification problems including three medical diagnosis problems (breast cancer, diabetes, and hepatitis) from the UCI Machine Learning repository (Prechelt, 1994). It is further shown that PSO-based algorithms can achieve a superior learning ability compared to other algorithms in terms of accuracy and speed. BP and PSO-training for MLPs has also been compared and evaluated over nonlinear function approximation in Kolen and Pollack (1990) and surprisingly it has been shown that PSO outperforms BP in terms of computation complexity required to achieve the same MSE level. In Sexton and Dorsey (2000), researchers performed a direct comparison of BP with the genetic algorithm (GA) for training ANNs over 10 real-world benchmark classification problems from Prechelt (1994). In this study, only three feed-forward network architectures, specifically single-hidden layer MLPs with 3, 6, and 12 hidden nodes, were used for all problems. According to the results, the GA performed consistently better than BP in terms of average classification error (CE) for all 10 problems. In a recent study (Mazurowski et al., 2008), two feed-forward ANN training methods, the traditional BP and PSO are compared by training a single hidden layer MLP with three hidden neurons on real clinical data for breast cancer diagnosis and in contradiction with the aforementioned results, it has been claimed that for imbalanced datasets, BP training yields better results than PSO in terms of average classification performance over the test set.

In this paper, in order to clarify such varying or even contradictory results of the previous studies, we first propose an accurate

and in-depth performance assessment approach for comparing local and global training methods, namely BP and PSO for MLPs, particularly over medical diagnosis problems. We shall particularly show that the major problem with the aforementioned evaluations is the usage of only one or few architecture(s) with a static training scheme for a given problem. In this case, one can find a unique architecture and a training depth for a particular problem, over which BP or PSO can surpass the other. In other words, it is quite evident that the performance of both BP and PSO may significantly vary with respect to the network architecture, the training depth and parameters used, and even the classification problem for which the training method is applied. Moreover, PSO and particularly BP yield significantly varying network parameters (weights and biases) after each training session and thus require an exhaustive number of training runs in order to determine statistically significant performance measures and accurate evaluations. In the current work, we focus especially on the average and the best performances that a particular training method can achieve whilst considering both training MSE and test classification error (CE) as the performance criteria. In the current assessment scheme, to avoid the bias of the network architecture used, rather than focusing only one or few architectures, we propose that the evaluations shall be performed over an architecture space containing a large variety of ANNs, from the simplest single-layer perceptrons (SLPs) to the MLPs with several hidden layers and neurons. Furthermore, the training depth is another important factor, which particularly affects the performance of local methods such as BP. For instance a shallow training (with a few iterations), BP is likely to yield a high training MSE and a low test classification error, and vice versa for a deep (or over-) training. As a global search method, PSO may or may not exhibit a similar phenomenon, depending on the other factors, such as the network architecture and training dataset used. Therefore, in this paper, both shallow and deep training depths shall be investigated distinctively while evaluating the performances of both methods in terms of training error (MSE over the training set), generalization capability (CE over the test set) and computational complexity level.

The rest of the paper is organized as follows. Section 2 surveys the related work on BP and PSO and their applications for training ANNs over medical diagnosis problems. Section 3 provides description of the proposed methodology to assess the performance of the corresponding local (BP) and global (PSO) methods for training feed-forward ANNs and discuss the results of the extensive set of experiments conducted over three benchmark problems from the field of medical diagnosis. Finally, Section 4 draws some conclusive remarks and discusses topics for future research.

2. Related work

2.1. The standard BP algorithm

Backpropagation (BP) (Rumelhart, Hinton, & Williams, 1986) is a well-known and widely used supervised training technique for multilayer ANNs with application to pattern recognition and classification in many areas including medical diagnostics. After the development of the BP training algorithm, multilayer perceptron networks have become the standard toolbox of neural network research. MLPs are feed-forward networks with one or more layers of nodes between the input and output nodes (Fig. 1). These additional (hidden) layers contain hidden neurons with nonlinear activation functions. Unlike single-layer perceptrons or multilayer nets with linear elements, a three-layer perceptron was proven to be capable of generating arbitrarily complex decision regions and computing any continuous likelihood function required in a classi-

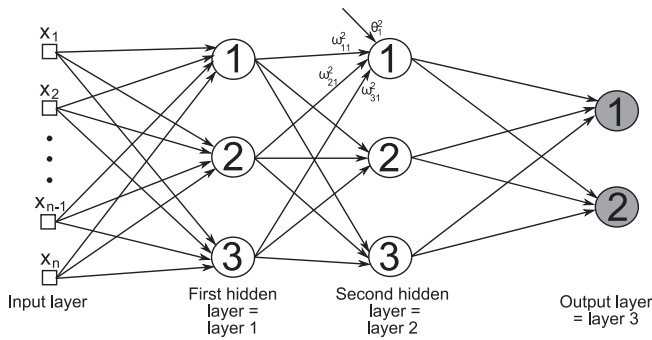


Fig. 1. A sample three-layer perceptron network with continuous valued inputs $\{x_1, \dots, x_n\}$, two outputs and two layers of hidden units.

fier (Lippmann, 1987). The BP algorithm may be viewed as a generalization of the least-mean-square (LMS) algorithm. It is an iterative gradient search technique designed to minimize the mean square error between the desired and actual net outputs. Backpropagation learning consists of two computational passes through the layers of an MLP: a forward pass which is a feed-forward propagation of input pattern signals through network, and a backward pass during which an error signal is computed at output units and propagated backwards through network. The standard BP algorithm can be summarized as follows:

1. Initialize the weights w_{jk}^l and biases θ_k^l randomly.
2. Feed pattern p to the network and compute the output $y_k^{p,l}$ of each neuron.
3. Calculate the error between the computed output $y_k^{p,o}$ of each output neuron and the desired output t_k^p as $e_k^{p,o} = t_k^p - y_k^{p,o}$.
4. For each neuron k , calculate the local gradients $\frac{\partial E^p}{\partial h_k^l}$ where E^p is the total error energy defined as $E^p = \frac{1}{2} \sum_{k \in o} (e_k^{p,o})^2$ and h_k^l is a uniform symbol for each parameter w_{jk}^l and θ_k^l . The name of the backpropagation algorithm comes from this, as it is necessary to start calculating the local gradients from the output layer and then recursively proceed backwards toward the input layer. Note that smooth (differentiable) nonlinear activation functions must be used for computing the local gradients. The formulas for calculating the local gradients at the output and hidden nodes of MLP can be found in Haykin (1999).
5. Update the parameters as follows:

$$h_k^l(t+1) = h_k^l(t) - \eta \frac{\partial E^p}{\partial h_k^l} \tag{1}$$

where η is the learning-rate parameter.

BP has the advantage of directed search, in that weights are always updated in such a way that minimizes the error. However, there are several aspects, which make the algorithm not guaranteed to be universally useful. Most troublesome is its strict dependency on a learning-rate parameter, which, if not set properly, can either lead to oscillation or indefinitely long training time. Network paralysis might also occur (Back & Tsoi, 1994), i.e. as the ANN trains, the weights tend to be quite large values and the training process can come to a virtual standstill. Furthermore, BP eventually slows down by an order of magnitude for every extra (hidden) layer added to ANN. Above all; BP is just a gradient descent algorithm in the error space, which can be complex and contain many deceiving local minima (multi-modal). Therefore, BP gets most likely trapped into a local minimum, making it entirely dependent on initial (weight) settings. There are many BP variants and extensions trying to address some or all of these problems such as (Hay-

kin, 1999; Joost & Schiffmann, 1998; Riedmiller & Braun, 1993); yet the performance and computational cost of each algorithm varies with respect to the problem at hand; and the question of which ANN architecture (number of layers and interconnections, number of nodes, etc.) should be used for a particular problem still remains unanswered. More detailed information about BP can be found in Chauvin and Rumelhart (1995).

2.2. The basic PSO algorithm

Particle swarm optimization (PSO) was introduced by Kennedy and Eberhart (1995) in 1995 as a population based stochastic search and optimization process. It is originated from a computer simulation of individuals (particles or living organisms) in a bird flock or a fish school (Wilson, 1975), which basically show a natural behavior when they search for some target (e.g. food). The goal is, therefore, to converge to the global optimum of some multi-dimensional and possibly nonlinear function or system. In principle, PSO follows the same path as other evolutionary algorithms (EAs) such as Genetic Algorithm (GA) (Goldberg, 1989), Genetic Programming (GP) (Koza, 1992), Evolutionary Strategies (ES) (Back & Kursawe, 1995), and Evolutionary Programming (EP) (Fayyad, Shapire, Smyth, & Uthurusamy, 1996). In a PSO process, a swarm of particles, each of which represents a potential solution in an optimization problem, navigates through the search space. Particles are initially distributed randomly over the search space and the goal is to converge to the global optimum of a function or a system. Each particle keeps track of its position in the search space and its best solution so far achieved. This is the personal best value (the so-called *pbest* in Kennedy and Eberhart (1995)) and the PSO process also keeps track of the global best solution so far achieved by the swarm with its particle index (the so-called *gbest* in Kennedy and Eberhart (1995)). During their journey with discrete time iterations, the velocity of each particle in the next iteration is affected by the best position of the swarm (best personal position of the particle *gbest* as the social component), the best personal position of the particle (*pbest* as the cognitive component, and its current velocity (the memory term). Both social and cognitive components contribute randomly to the position of the particle in the next iteration. As a stochastic search algorithm in a multi-dimensional (MD) search space, PSO exhibits some major shortcomings similar to the other EAs. A crucial problem for PSO is that parameter variations may result in large performance shifts (Lovberg & Krink, 2002). The second shortcoming is due to the direct link of information flow between particles and *gbest*, which then "guides" the rest of the swarm and thus resulting in the creation of "similar" particles with a loss of diversity. Hence this phenomenon increases the probability of being trapped in local optima (Riget & Vesterstrom, 2002) and it is the main source of premature convergence especially when the search space is in high dimensions (Van den Bergh, 2002) and the problem to be optimized is multi-modal (Riget et al., 2002). Another reason for the premature convergence is that particles are flown through a single point which is (randomly) determined by *gbest* and *pbest* positions and this point is not even guaranteed to be a local optimum (Van den Bergh, 2002). Various modifications and PSO variants have been proposed in order to address this problem such as (Christopher & Seppi, 2004; Clerc, 1999; Higashi & Iba, 2003; Lovberg, 2002; Riget et al., 2002; Shi & Eberhart, 1998).

In the basic PSO method, a swarm of particles fly through an N -dimensional search space where each particle represents a potential solution to the optimization problem. Each particle a in the swarm $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, is represented by the following characteristics:

$x_{aj}(t)$	j th dimensional component of the position of particle a , at time t
$v_{aj}(t)$	j th dimensional component of the velocity of particle a , at time t
$y_{aj}(t)$	j th dimensional component of the personal best ($pbest$) position of particle a , at time t
$\hat{y}_j(t)$	j th dimensional component of the global best ($gbest$) position of swarm, at time t

Let f denote the fitness function to be optimized. Without loss of generality assume that the objective is to find the minimum of f in N -dimensional space. Then the personal best of particle a can be updated in iteration $t + 1$ as,

$$y_{aj}(t+1) = \begin{cases} y_{aj}(t) & \text{if } f(x_a(t+1)) > f(y_a(t)) \\ x_{aj}(t+1) & \text{else} \end{cases} \quad j = 1, 2, \dots, N \quad (2)$$

Since $gbest$ is the index of the global best (GB) particle, then $\hat{y}(t) = y_{gbest}(t) = \arg \min_{i \in [1, S]} (f(y_i(t)))$. Then for each iteration in a PSO process, positional updates are performed for each dimension component, $j \in [1, N]$ and for each particle index, $a \in [1, S]$, as follows:

$$\begin{aligned} v_{aj}(t+1) &= w(t)v_{aj}(t) + c_1r_{1j}(t)(y_{aj}(t) - x_{aj}(t)) + c_2r_{2j}(t)(\hat{y}_j(t) - x_{aj}(t)) \\ x_{aj}(t+1) &= x_{aj}(t) + v_{aj}(t+1) \end{aligned} \quad (3)$$

where $w(t)$ is the inertia weight and c_1, c_2 are the acceleration constants which are initially set to 2 (Shi and Eberhart, 1998). $r_{1j} \sim U(0, 1)$ and $r_{2j} \sim U(0, 1)$ are random variables with uniform distribution. Although the use of inertia weight, $w(t)$, was later added by Shi and Eberhart (1998), into the velocity update equation, it is widely accepted as the basic form of PSO algorithm. A larger value of $w(t)$ favors exploration while a small inertia weight favors exploitation. As originally introduced, $w(t)$ is often linearly decreased from a high value (e.g. 0.9) to a low value (e.g. 0.4) during iterations of a PSO run, which updates the positions of the particles using Eq. (3). Depending on the problem at hand, PSO iterations can be repeated until a specified number of iterations, say $IterNo$, is exceeded, velocity updates become zero, or the desired fitness score is achieved (i.e. $f < \epsilon_c$). Accordingly, the general pseudo-code of the PSO is presented in Table 1.

2.3. Applications of BP and PSO over ANNs

For the purpose of medical diagnostics, ANNs have been proposed as decision support tools and have been successfully applied to a wide range of problems such as automated ECG signal diagnosis (Hosseini et al., 2006), electroencephalogram (EEG) waveform classification (Haselsteiner & Pfurtscheller, 2000), decision support systems for breast cancer diagnosis (Lisboa & Taktak, 2006), diagnosis of acute myocardial infarction (coronary occlusion) (Baxt, 1990), recognition of tumors in ultrasound images of the eye (Silverman & Noetzel, 1990), diagnosis of acute pulmonary embolism (Tourassi, Floyd, Sostman, & Coleman, 1993), and differential diagnosis of six dermatology (erythematous-squamous) diseases (West & West, 2000). MLPs trained with the BP algorithm are used in the majority of the techniques proposed for medical decision support systems where the primary purpose is to augment the physician's decision making in a disease or anomaly diagnosis. However, there are several issues that should be addressed when applying ANN-based classifiers in the medical domain. Besides the selection of the network architecture (type of network, number of layers and interconnections, number of nodes, etc.) and settings of training parameters, the effects of limited training data, description of the input space, large number of features, and class imbal-

Table 1
Pseudo-code for PSO algorithm.

<p>PSO (termination criteria: $\{IterNo, \epsilon_c, \dots\}, V_{max}$)</p> <ol style="list-style-type: none"> 1. For $\forall a \in [1, S]$ do: <ol style="list-style-type: none"> 1.1. Randomize $x_a(1), v_a(1)$ 1.2. Let $y_a(0) = x_a(1)$ 1.3. Let $\hat{y}(0) = x_a(1)$ 2. End For. 3. For $\forall t \in [1, IterNo]$ do: <ol style="list-style-type: none"> 3.1. For $\forall a \in [1, S]$ do: <ol style="list-style-type: none"> 3.1.1. Compute $y_a(t)$ using Eq. (1) 3.1.2. If $(f(y_a(t)) < \min_{1 \leq i < a} (f(\hat{y}_i(t-1))))$ then $gbest = a$ and $\hat{y}(t) = y_a(t)$ 3.2. End For. 3.3. If any termination criterion is met, then Stop. 3.4. For $\forall a \in [1, S]$ do: <ol style="list-style-type: none"> 3.4.1. For $\forall j \in [1, N]$ do: <ol style="list-style-type: none"> 3.4.1.1. Compute $v_{aj}(t+1)$ using Eq. (2) 3.4.1.2. If $(v_{aj}(t+1) > V_{max})$ then clamp it to $v_{aj}(t+1) = V_{max}$ 3.4.1.3. Compute $x_{aj}(t+1)$ using Eq. (2) 3.4.2. End For. 3.5. End For. 4. End For.
--

ance should be considered during the development phase and the subsequent performance evaluation. Interestingly, despite the fact that many new studies using neural network classifiers have recently been proposed for medical decision making with promising results, only few techniques were able to find their way into clinical use mainly due to poor benchmarking and especially the lack of a proper assessment methodology (Lisboa & Taktak, 2006). One important implication is the need of more extensive and rigorous methodologies for evaluating neural network systems used in medical diagnosis.

Several researchers have successfully applied PSO to train feed-forward and recurrent ANNs that are used in classification problems, some of which are from the medical field. One of the advantages of PSO-based neural network training is that any proper objective function which is more relevant to a particular problem can be chosen and PSO is applicable to any type of ANN, with any activation function. In Eberhart and Hu (1999), a PSO was applied to evolve (train) a feed-forward neural network, which would distinguish between normal subjects and those with tremor (Parkinson's disease or essential tremor) and encouraging results were obtained. Another comparative study over MLPs, which investigates training methods based on many PSO variants, such as multi-start PSO, guaranteed convergence PSO, the traditional BP method, and GA-based techniques was presented in Van den Bergh (2002). This study shows that over the three medical datasets (*breast cancer, diabetes, and hepatitis*) from *Proben1* (Prechelt, 1994), PSO and its variants achieve a superior training performance compared to the others in terms of classification accuracy on the test set and the training speed. Recently, the effect of class imbalance in the training dataset over the performance of feed-forward ANN classifiers for medical diagnosis was investigated in Mazurowski et al. (2008). As in Van den Bergh (2002), a feed-forward ANN with a single-hidden layer with three neurons was used and BP and PSO were both employed to evaluate the classifier performance using a clinical dataset for breast cancer diagnosis. However, contrary to Van den Bergh (2002), BP training was found to be superior to PSO in terms of (average) test performance.

As one of the most widely applied ANNs, in this study we shall focus on the (supervised) training process of MLP networks. Let N_h^l be the number of hidden neurons in layer l of a MLP with input and output layer sizes N_I and N_O , respectively. The input neurons are merely fan-out units since no processing takes place. Let F be the

activation function applied over the weighted inputs plus a bias, as follows:

$$y_k^{p,l} = F(s_k^{p,l}) \quad \text{where } s_k^{p,l} = \sum_j w_{jk}^{l-1} y_j^{p,l-1} + \theta_k^l \quad (4)$$

where $y_k^{p,l}$ is the output of the k th neuron of the l th hidden/output layer when the pattern p is fed, w_{jk}^{l-1} is the weight from the j th neuron in layer $l - 1$ to the k th neuron in layer l , and θ_k^l is the bias value of the k th neuron of the l th hidden/output layer. The training mean squared error, MSE, is formulated as:

$$MSE = \frac{1}{2PN_0} \sum_{p \in T} \sum_{k=1}^{N_0} (t_k^p - y_k^{p,O})^2 \quad (5)$$

where t_k^p is the target (desired) output and $y_k^{p,O}$ is the actual output from the k th neuron in the output layer, $l = O$, for pattern p in the training set T with size P . The error measure of MSE is set as an objective (fitness) function of both BP and PSO-training methods to assess their performance under equal training conditions. In PSO, each particle a in the swarm, $\zeta = \{x_1, \dots, x_a, \dots, x_S\}$, has the positional component formed as, $x_{a,j}(t) = \left\{ \left\{ w_{jk}^0 \right\}, \left\{ w_{jk}^1 \right\}, \left\{ \theta_k^1 \right\}, \left\{ w_{jk}^2 \right\}, \left\{ \theta_k^2 \right\}, \dots, \left\{ w_{jk}^{O-1} \right\}, \left\{ \theta_k^{O-1} \right\}, \left\{ \theta_k^O \right\} \right\}$ where $\left\{ w_{jk}^l \right\}$ and $\left\{ \theta_k^l \right\}$ represent a potential solution to the problem (the set of weights and biases of layer l). Note that the input layer ($l = 0$) contains only weights whereas the output layer ($l = O$) has only biases. Setting MSE in Eq. (5) as the fitness function enables PSO to perform evolutions of each network parameters within its native process and converge to the best global position found during all iterations.

3. BP vs. PSO: comparative performance evaluation over medical data

3.1. The experimental setup

The architecture space can be defined over a wide range of configurations, i.e. say from a single-layer perceptron (SLP) to complex MLPs with many hidden layers. Suppose, for the sake of simplicity, a range is defined for the number of layers, $[L_{\min}, L_{\max}]$ and another for the number of neurons for each hidden layer l , $[N_{\min}^l, N_{\max}^l]$. Without loss of generality, assume that the size of both input and output layers is determined by the problem and hence fixed. Consequently, the architecture space can now be defined by only two (MLP) configuration sets, $R_{\min} = \{N_I, N_{\min}^1, \dots, N_{\min}^{L_{\max}-1}, N_O\}$ and $R_{\max} = \{N_I, N_{\max}^1, \dots, N_{\max}^{L_{\max}-1}, N_O\}$, one for minimum and the other for maximum number of neurons allowed for each layer of a MLP. The size of both arrays is naturally $L_{\max} + 1$ where corresponding entries define the range of neurons possible on the l th hidden layer for all those MLPs, which can have an l th hidden layer. The size of input and output layers, $\{N_I, N_O\}$, is fixed and remains the same for all configurations in the architecture space within which any l -layer MLP can be defined providing that $L_{\min} \leq l \leq L_{\max}$. $L_{\min} \geq 1$ and L_{\max} can be set to any value meaningful for the problem at hand. In this way, all network configurations in the architecture space are enumerated into a hash table with a proper hash function, which basically ranks the networks with respect to their complexity, i.e. associates higher hash indices to networks with higher complexity. The hash function then enumerates all potential MLP configurations into hash indices, starting from the simplest MLP with $L_{\min} - 1$ hidden layers, each of which has a minimum number of neurons given in R_{\min} , to the most complex network with $L_{\max} - 1$ hidden layers, each of which has a maximum number of neurons given in R_{\max} . Take, for instance, the following configuration sets, $R_{\min} = \{9, 1, 1, 2\}$ and $R_{\max} = \{9, 8, 4, 2\}$, which indicate that $L_{\max} = 3$. If $L_{\min} = 1$ then the hash function enu-

Table 2

A sample architecture space for MLP configuration sets $R_{\min} = \{9, 1, 1, 2\}$ and $R_{\max} = \{9, 8, 4, 2\}$.

Index	Configuration	Index	Configuration
0	9 × 2	21	9 × 5 × 2 × 2
1	9 × 1 × 2	22	9 × 6 × 2 × 2
2	9 × 2 × 2	23	9 × 7 × 2 × 2
3	9 × 3 × 2	24	9 × 8 × 2 × 2
4	9 × 4 × 2	25	9 × 1 × 3 × 2
5	9 × 5 × 2	26	9 × 2 × 3 × 2
6	9 × 6 × 2	27	9 × 3 × 3 × 2
7	9 × 7 × 2	28	9 × 4 × 3 × 2
8	9 × 8 × 2	29	9 × 5 × 3 × 2
9	9 × 1 × 1 × 2	30	9 × 6 × 3 × 2
10	9 × 2 × 1 × 2	31	9 × 7 × 3 × 2
11	9 × 3 × 1 × 2	32	9 × 8 × 3 × 2
12	9 × 4 × 1 × 2	33	9 × 1 × 4 × 2
13	9 × 5 × 1 × 2	34	9 × 2 × 4 × 2
14	9 × 6 × 1 × 2	35	9 × 3 × 4 × 2
15	9 × 7 × 1 × 2	36	9 × 4 × 4 × 2
16	9 × 8 × 1 × 2	37	9 × 5 × 4 × 2
17	9 × 1 × 2 × 2	38	9 × 6 × 4 × 2
18	9 × 2 × 2 × 2	39	9 × 7 × 4 × 2
19	9 × 3 × 2 × 2	40	9 × 8 × 4 × 2
20	9 × 4 × 2 × 2		

merates all MLP configurations in the architecture space as shown in Table 2. Note that in this example, the input and output layer sizes are 9 and 2, which are eventually fixed for all MLP configurations. The hash function associates the first index ($d = 0$) with the simplest possible architecture, i.e., a SLP with only input and output layers (9×2). From indices 1 to 8, all configurations belong to 2-layer MLP with a single-hidden layer containing a varying number of neurons between 1 and 8 (as specified in the 2nd entries of arrays R_{\min} and R_{\max}). Similarly, for indices 9 and up, 3-layer MLPs are enumerated in which the number of neurons in the 1st and 2nd hidden layers sizes are varied according to the corresponding entries in R_{\min} and R_{\max} . Finally, the most complex MLP with the largest number of possible layers and the highest number of neurons is associated with the highest index, $d = 40$. Therefore, all 41 entries in the hash table span the architecture space with respect to the configuration complexity.

The comparative evaluations of both training algorithms were performed using medical diagnosis benchmark dataset from the UCI Machine Learning repository (Prechelt, 1994), which is partitioned into three sets: training, validation and testing. There are several techniques (Amari, Murata, Muller, Finke, & Yang, 1997) to use training and validation sets individually to prevent over-fitting and thus to improve the classification performance in the test data. However, there is no universally effective technique and there are several research articles reporting against the use of the cross-validation technique in the design and training of MLP networks (Amari et al., 1997; Andersen & Martinez, 1999). In this study, for simplicity and to obtain an unbiased performance measure under equal training conditions, the validation and training sets are simply combined to be used for training. From Proben1 repository (Prechelt, 1994), three benchmark classification problems, breast cancer, heart disease and diabetes, are selected, which were commonly used by the previous studies. These are medical diagnosis problems, which present the following attributes:

- All of them are real-world problems based on medical data from human patients.
- The input and output attributes are similar to those used by a medical doctor.
- Since medical examples are expensive to get, the training sets are quite limited.

We now briefly describe each classification problem next.

3.1.1. Breast cancer

The objective of this data set is to classify breast lumps as either benign or malignant according to microscopic examination of cells that are collected by needle aspiration. There are 699 exemplars of which 458 are benign and 241 are malignant and they are originally partitioned as 350 for training, 175 for validation and 174 for testing. The data set consists of nine input attributes and two output attributes, i.e. each input pattern is described by 9-dimensional vector and there are two possible outcomes of the classifier. It is created at University of Wisconsin Madison by Dr. William Wolberg.

3.1.2. Heart disease

The initial data set consists of 920 exemplars with 35 input attributes, some of which are severely missing. Hence a second data set is composed using the cleanest part of the preceding set, which was created at Cleveland Clinic Foundation by Dr. Robert Detrano. The Cleveland data is called "heartc" in Proben1 repository and contains 303 exemplars but six of them still contain missing data and are hence discarded. The remaining exemplars are partitioned as follows: 149 for training, 74 for validation and 74 for testing. There are 13 input and 2 output attributes. The purpose is to predict the presence of a heart disease according to the input attributes.

3.1.3. Diabetes

This dataset is used to predict diabetes diagnosis among Pima Indians. The data is collected from female patients, aged 21 years or older. There are total of 768 exemplars of which 500 are classified as diabetes negative and 268 as diabetes positive. The data set is originally partitioned as 384 for training, 192 for validation and 192 for testing. It consists of eight input attributes and two output attributes.

The input attributes of all data sets are scaled within the range [0, 1] by a linear function. Their output attributes are encoded using a 1-of- c representation using c classes. The winner-takes-all methodology is applied so that the output of the highest activation designates the class. Overall, our experimental setup becomes identical to those used in the previous studies and thus fair com-

parative evaluations can now be made over the classification error rate of the test data. In all experiments in this section we use the sample architecture space given in Table 2, which has the generalized form as, $R_{\min} = \{N_i, 1, 1, N_o\}$ and $R_{\max} = \{N_i, 8, 4, N_o\}$ containing the compact 1-, 2- or 3-layer MLPs where N_i and N_o , are determined by the number of input and output attributes of the classification problem. For BP, all networks were trained with 500 (shallow training) and with 5000 (deep training) iterations with a low learning rate of 0.02. For PSO-training, in addition to default settings for the standard algorithm parameters as defined in Section 2.3, the number of particles was set to 40 ($S = 40$) and the number of training iterations was set to 200 for shallow and 2000 for deep training case. For all experiments in this section, unless stated otherwise, 100 independent runs are performed for each configuration to compute the error statistics plots for each dataset. We mainly consider two major criteria for the performance assessment: (1) training MSE, which indicates the error minimization achieved by each method and (2) test CE, which is the primary objective of the classifier as it shows the classification accuracy level achieved as well as the generalization capability of each method. Using the corresponding error statistics plots, both criteria shall then be statistically evaluated by considering *on the average* (i.e. mean MSE and CE) and *the best* (i.e. minimum MSE and CE) performances achieved by each method, BP and PSO.

3.2. Evaluations of results with the proposed methodology

In order to perform a comprehensive and systematic assessment of predictive performance of ANN classifiers in medical diagnosis, we apply *exhaustive* BP- and PSO-training for each network configuration in the architecture space, which is defined over MLPs with sigmoid activation functions. In this way we can escape from the bias or possible effect of a particular network over the performance, which was the case of many of the aforementioned studies that were mostly performed using only one or few fixed network architecture(s). Furthermore, to assess the effect of the training depth on both BP and PSO, both shallow and deep training will be applied over every network configuration in the architecture space by setting the number of iterations appropriately.

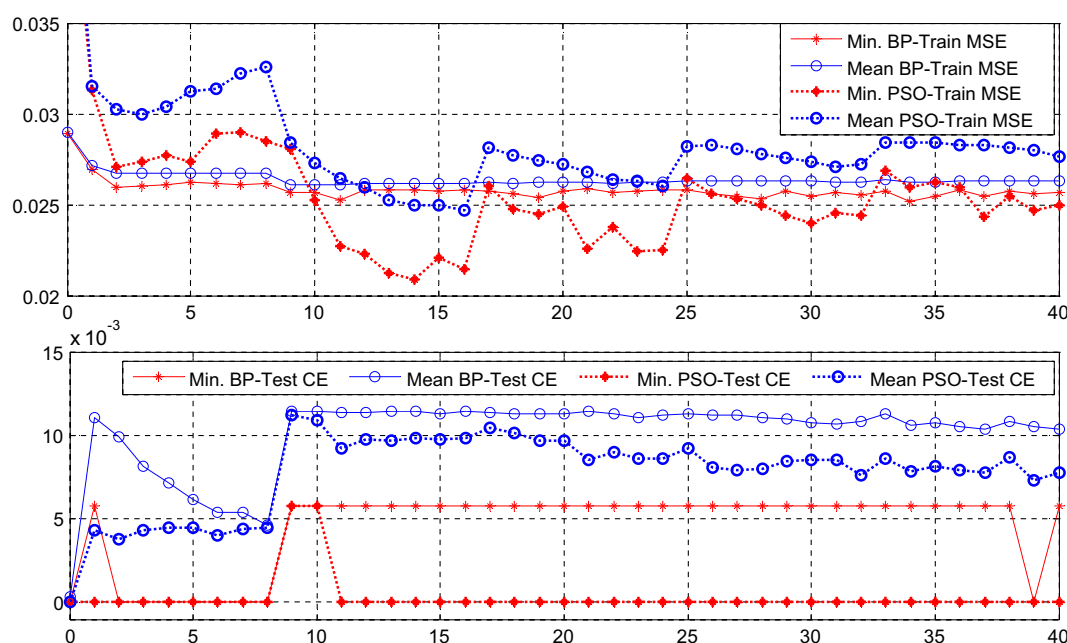


Fig. 2. Train (top) and test (bottom) error statistics vs. hash index plots from shallow BP- and PSO-training over the breast cancer dataset.

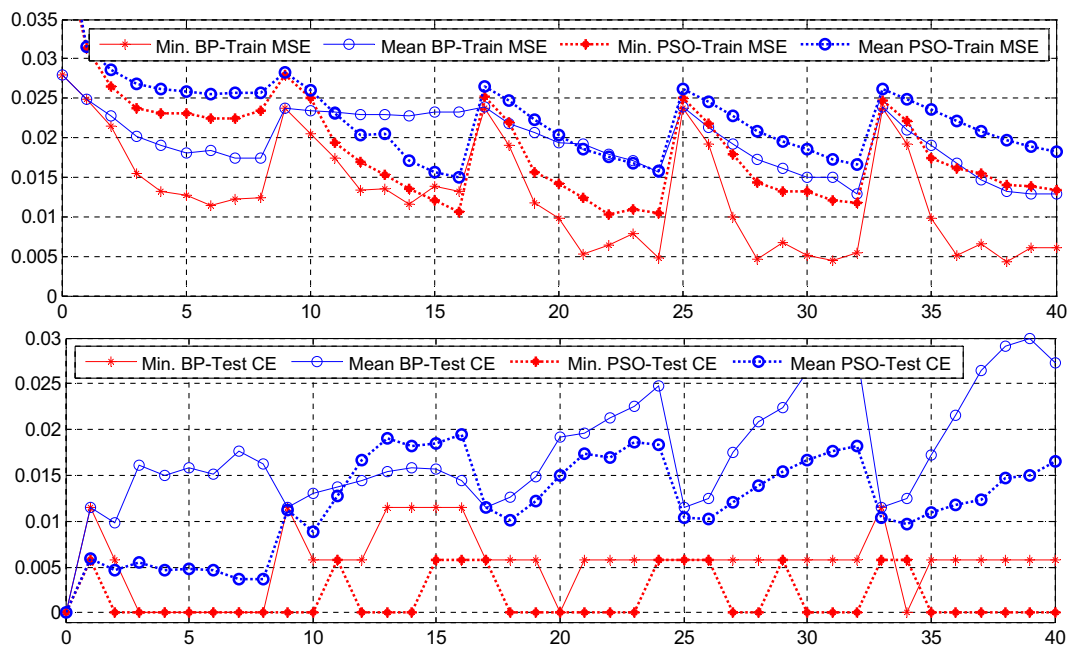


Fig. 3. Train (top) and test (bottom) error statistics vs. hash index plots from deep BP- and PSO-training over the breast cancer dataset.

Fig. 2 presents the corresponding error statistics plots from the shallow training over the breast cancer dataset. BP in general achieves the lowest average training MSEs within a narrow variance except few network configurations with the corresponding indices, $d \in [13, 16]$ where PSO slightly surpasses BP. On the other hand, PSO achieves the best training performances (i.e. minimum MSEs) over the majority of network configurations except for the compact ones ($d \in [0, 9]$) where BP is consistently more successful. The lowest overall training MSE (both average and minimum) is too achieved by PSO using the configuration with the hash indices $d = 14$ and $d = 16$ (MLPs: $9 \times 6 \times 1 \times 2$ and $9 \times 8 \times 1 \times 2$), respectively. In terms of the classification performances over the test set, the results are consistently in favor of PSO, which performs better than BP with respect to both performance criteria. Particularly, PSO achieved the optimal 0% CE (i.e.100% classification accuracy) as its best performance among all networks except for only two networks ($d = 9$ and $d = 10$), whereas BP managed to achieve this only over the compact networks (i.e. $d = 0$ and $d \in [2, 8]$), plus the complex MLP with $d = 39$. Overall, PSO usually demonstrates a better classification performance for the breast cancer dataset with the shallow training.

The error statistics plots obtained from deep training of all networks in the architecture space by both methods are shown in Fig. 3. In this case, both BP and PSO achieve lower training MSEs as a natural consequence of the deep- or over-training, and BP in general achieves the lowest average training MSEs, particularly on complex networks with two hidden layers but it also surpasses PSO in terms of the minimum training MSEs except for only few networks. Due to such over-training, the classification performance of both methods is expected to degrade, which is the case as shown by the bottom plots of Fig. 3. However, the degradation on PSO's performance is not as severe as that of BP. Furthermore, note that there is almost no performance loss in the case of compact networks, due to PSO's global search ability. Particularly for complex networks, a significant performance gap in terms of average test CE occurs between the two methods in favor of PSO since BP, as a deterministic local search method, is drastically affected by the over-fitting of the training data and thus exhibits significantly worse average classification performance. This is somewhat true

for PSO too; especially its minimum CE cannot anymore guarantee 0% CE level for all network configurations.

Fig. 4 presents the corresponding error statistics plots from the shallow training over the heart disease dataset. Both average and minimum training MSE statistics of both methods vary significantly with respect to the network configuration, making either method better than the other for a given network and error criterion. This is a good example that clearly shows the effect of different network configurations over the performance of each method. Therefore, as discussed earlier about the aforementioned studies in this field, using only one or few architectures for comparative evaluations may favor either method and hence such a static and limited approach is not sufficient to draw reliable conclusions. The lowest training MSE (both average and minimum) is achieved by PSO using the network with the hash index $d = 16$ (MLP: $13 \times 8 \times 1 \times 2$). As in the previous dataset, about the classification performances over the test set, the results are consistently in favor of PSO, which performs better than BP for all networks in the architecture space with respect to both performance criteria.

On contrary to shallow training results, the top plot in Fig. 5 indicates that whenever deep training is performed over this dataset, BP surpasses PSO with respect to the training MSEs (both average and minimum) for all networks. Hence due to the over-fitting of training data, the classification performance of BP over the test set is quite degraded whilst no significant performance degradation occurs for PSO. PSO, once again, exhibits its immunity against over-training due to its global search ability and proves to yield the best classification performance over the test set (i.e. well generalization) regardless of the training depth. This is true for both average and the best performance criteria considered (see red¹ and blue curves of the bottom plot in Fig. 5). PSO achieves the overall best classification performance, ~13% CE, from the three different networks with the corresponding hash indices, $d = 14, 30$ and 39 although no network configuration makes too much difference when on the average classification performance is concerned.

¹ For interpretation of color in Fig. 5, the reader is referred to the web version of this article.

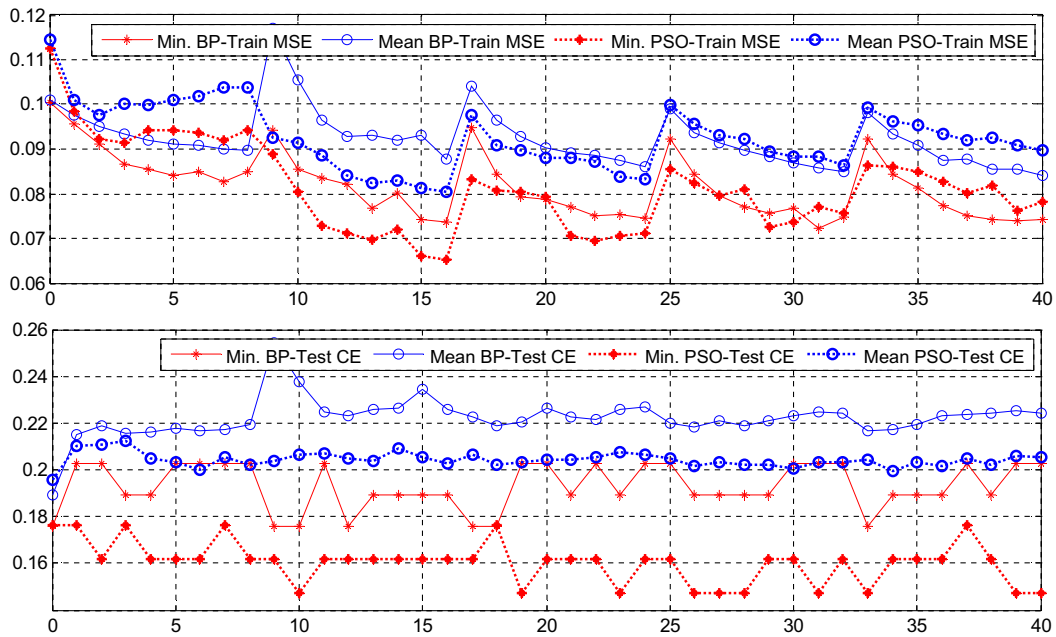


Fig. 4. Train (top) and test (bottom) error statistics vs. hash index plots from shallow BP- and PSO-training over the heart disease dataset.

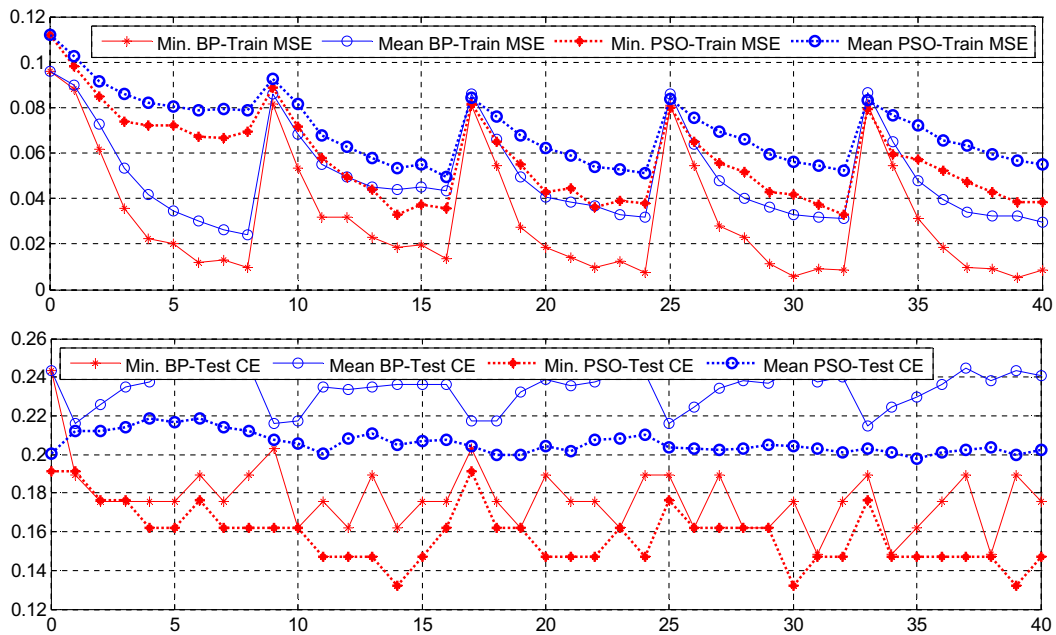


Fig. 5. Train (top) and test (bottom) error statistics vs. hash index plots from deep BP- and PSO-training over the heart disease dataset.

Fig. 6 presents the corresponding error statistics plots from the shallow training over the diabetes dataset. Similar comments can be made about the training performance of PSO and BP as in the shallow training experiments over the heart disease dataset, i.e. although BP is consistently better than PSO for compact networks, their training performances (minimum and average MSEs) are quite comparable and varying along with the network configuration. In terms of the classification performance over the test set, PSO usually achieves slightly lower CEs but the results are again quite comparable. When the minimum CEs are concerned, from the network with the hash index $d = 16$ (MLP: $8 \times 8 \times 1 \times 2$) PSO achieved a minimum of 17.1% CE that is slightly lower than the

18.8% minimum CE achieved by BP from the network with the hash index $d = 4$ (MLP: $8 \times 4 \times 2$). Finally, according to the error statistics plots in Fig. 7 obtained by deep training over the same dataset, similar comments can be made about both training (MSE) and generalization (test CE) performances of PSO and BP as in the deep training experiments over the heart disease dataset, i.e. PSO yields almost the same average training MSE levels and BP significantly reduces both average and minimum MSE levels, as expected. One observation worth mentioning here is that the training and test performances of BP in both deep and shallow training exhibits a large variation with respect to the network configuration used (e.g. compare for instance the mean BP training MSE or test CE

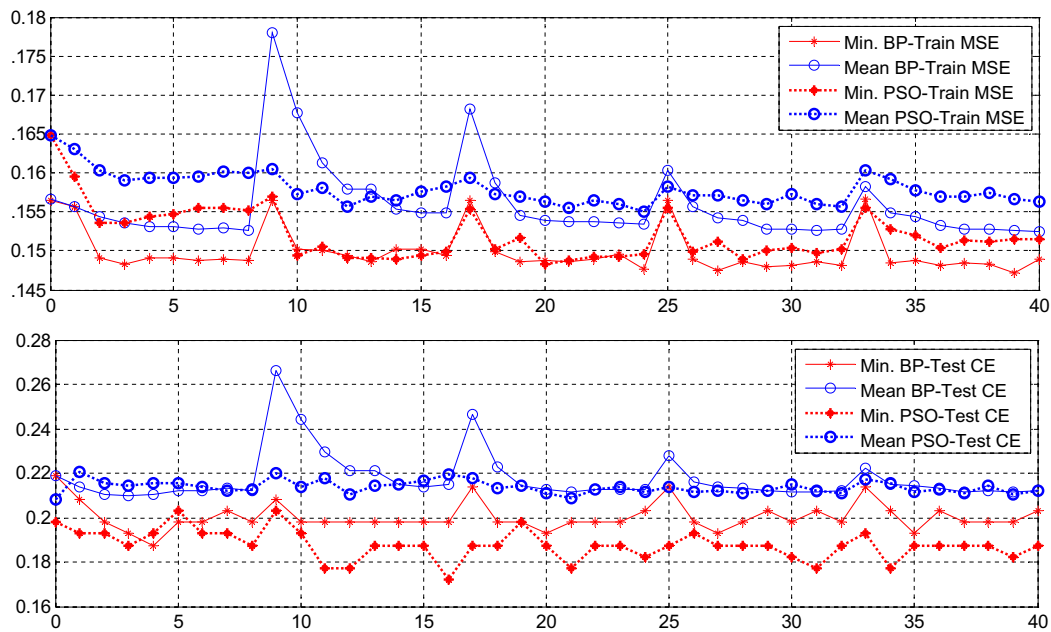


Fig. 6. Train (top) and test (bottom) error statistics vs. hash index plots from shallow BP- and PSO-training over the diabetes dataset.



Fig. 7. Train (top) and test (bottom) error statistics vs. hash index plots from deep BP- and PSO-training over the diabetes dataset.

for $d = 8$ and $d = 9$), whereas the corresponding performance levels of PSO are more stable and usually with a smaller variance, regardless of the network configuration.

The overall test CE statistics of both training techniques (BP and PSO) computed over all configurations in selected MLP architecture spaces for each dataset are enlisted in Table 3. We used the following three statistics: minimum min , mean μ , and standard deviation σ , respectively, which are computed per training depth (deep and shallow) and separately for the *minimum* and the *mean* test CEs. For all datasets, *minimum* test CE statistics in the table clearly indicates that the overall best classification performances are achieved by PSO-training independent from the training depth. This is also

true for the average classification performances with the only exception that in the *heart disease* dataset, minimum of the mean test CEs (corresponding to the best “on the average” test performance) is achieved by BP with the shallow training.

Finally, in order to accomplish the comparative performance evaluations of each method with respect to the variations in the training depth, we have selected a particular network configuration with the hash index $d = 16$, which is a relatively compact and one of the best-performing classifier configuration within the sample architecture space, and we have performed exhaustive training (with 100 runs) for each of the 10 intermediate (training) depths, between the corresponding shallow and deep training, i.e.

Table 3
The overall test CE statistics. The minimum CE statistics are highlighted.

Data set	Training method	Training depth	Min. test CE statistics			Mean test CE statistics		
			<i>min</i>	μ	σ	<i>min</i>	μ	σ
Breast cancer	BP	Shallow	0	0.0045	0.0024	0.0003	0.0101	0.0024
	PSO	Shallow	0	0.0003	0.0013	0	0.0078	0.0024
	BP	Deep	0	0.0055	0.0036	0	0.0176	0.0064
	PSO	Deep	0	0.0015	0.0026	0	0.0121	0.0052
Diabetes	BP	Shallow	0.1875	0.2002	0.0063	0.2101	0.2175	0.0112
	PSO	Shallow	0.1719	0.1878	0.0067	0.2079	0.2137	0.0028
	BP	Deep	0.1719	0.1941	0.0129	0.2135	0.2246	0.0068
	PSO	Deep	0.1667	0.1836	0.0101	0.2069	0.2135	0.0040
Heart disease	BP	Shallow	0.1757	0.1928	0.0100	0.1893	0.2222	0.0087
	PSO	Shallow	0.1471	0.1603	0.0092	0.1957	0.2043	0.0031
	BP	Deep	0.1486	0.1773	0.0171	0.2150	0.2340	0.0096
	PSO	Deep	0.1324	0.1578	0.0151	0.1976	0.2060	0.0054

[500, 5000] for BP, and [200, 2000] for PSO. Fig. 8 shows the training MSE and test CE plots vs. training depth for all three medical problems. From the figure, it is clear that both methods reduce training MSE with increasing training depths, as a natural consequence of over-fitting of the training data. On the other hand, PSO achieves lower average and minimum training MSEs for the *breast cancer*, higher for the diabetes and quite similar for the *heart disease* datasets, respectively. The classification performance of PSO shows a strong immunity against variations in training depth and it generally achieves the lowest minimum CEs. For this particular network, either BP or PSO can achieve a better average classification performance than the other, depending on the training depth. Hence this clearly draws the conclusion that the training

depth too should be considered while comparing and/or analyzing individual performance of each method.

3.3. Discussion and computational complexity analysis

The overall experimental results, first of all, show that the performance of both methods, BP and PSO, directly depends on the network architecture used and the training depth applied, each of which has varying effects on the two performance criteria employed, minimum and average training MSE and test CE. The former basically shows the search or optimization performance of the method in the training set whereas the latter signifies the generalization ability and the main objective of any ANN training

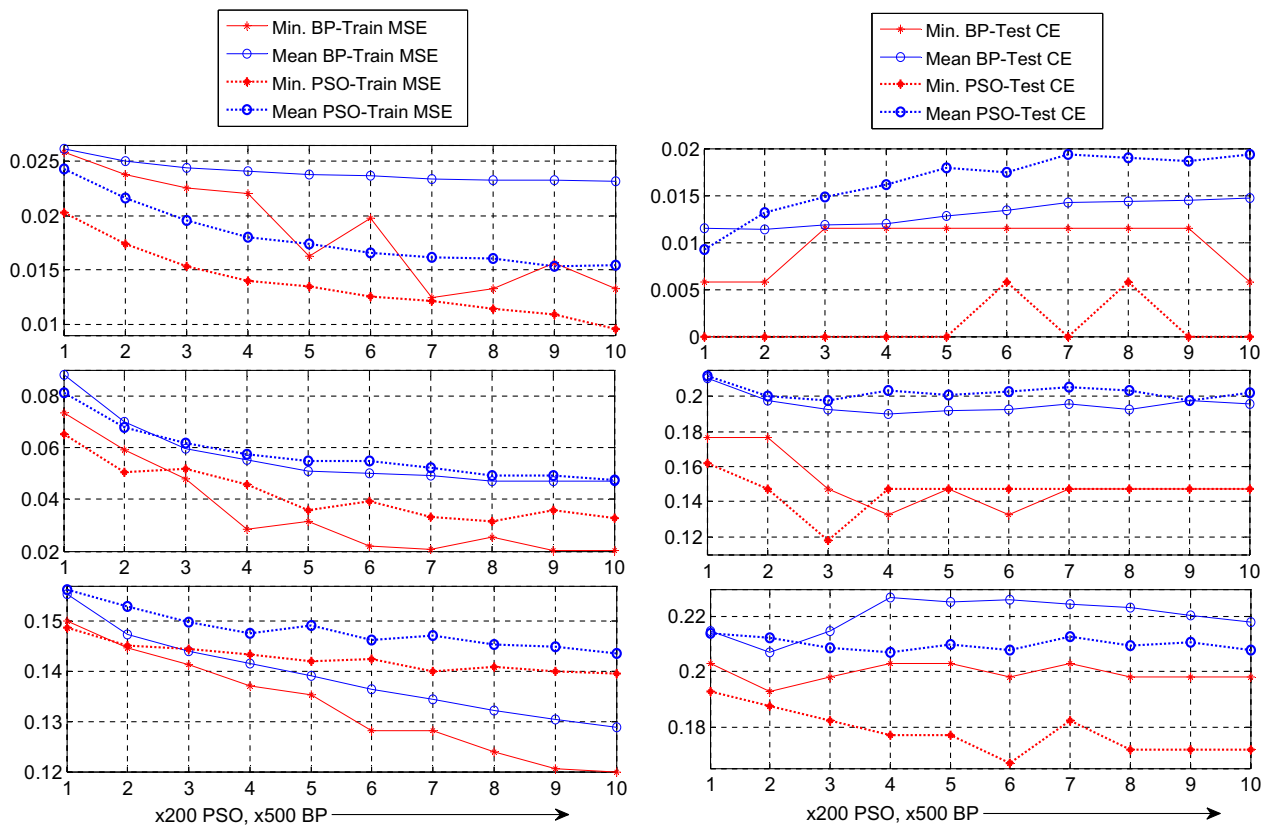


Fig. 8. Error statistics (for network configuration with the hash index $d = 16$) vs. training depth plots using BP and PSO over the *breast cancer* (top), *heart disease* (middle), and *diabetes* (bottom) datasets.

Table 4

Average processing times (in milliseconds) of the BP and PSO-training algorithms over the feed-forward architecture space for three medical problems.

Training	Breast cancer		Heart disease		Diabetes	
	BP	PSO	BP	PSO	BP	PSO
Shallow	49.0	236.6	21.3	208.2	114.0	391.3
Deep	436.5	2272.0	217.9	1702.8	670.4	2510.4

method, i.e. the classification performance within the test set. The results justify the use of the proposed assessment technique for both methods, BP and PSO, over the medical datasets. Otherwise, it would be inevitable to obtain varying or even worse contradicting and biased results, which make them all unreliable. For example in Sexton and Dorsey (2000), it is evident that the performance of BP method is significantly underestimated, i.e. the authors of Sexton and Dorsey (2000) reported mean test CE as 3.01% for the breast cancer, 24.89% for the heart disease, and 29.62% for the diabetes datasets where they used only three single-hidden layer MLPs with 3, 6, and 12 hidden nodes, and performed only 10 training runs. In another study (Mazurowski et al., 2008), the authors performed comparative evaluations between the traditional BP and PSO methods over a breast cancer dataset and by using only a single-hidden layer MLP with three hidden neurons. With such a limited approach, they made a general conclusion that BP performs better than PSO for imbalanced medical datasets. BP perhaps may surpass PSO on that particular MLP, for that dataset and under those specific conditions they employed; however, it is demonstrated in the current work that such an approach is nevertheless not sufficient to draw such a general and decisive conclusion.

Finally, we shall perform computational complexity analysis, which mainly depends on some common properties such as the size of the input and output layers along with the size of the training set (problem specific), and the total number of connections in the network (network complexity). All experiments in this section are performed on a computer with P-IV 3 GHz CPU and 1 GB RAM. BP method consists of one forward and one backward propagation, which requires computation of the error at the output nodes and back-propagating it to the hidden nodes in each hidden layer by calculating the error derivative with respect to weights. Thus the backpropagation of the error (MSE) is computationally the most expensive operation, (it was noticed that backward propagation is 4–8 times more computationally expensive than forward propagation). On the other hand, during a PSO process, at each iteration and for each particle, first the network parameters are extracted from the particle and input vectors are only forward propagated to compute the average error (MSE) at the output layer. Let T be the size of the training data set, S be the swarm size, E be the total number of iterations (epochs) in a PSO run, and μ_t be the average time for a forward propagation. Since the computational complexity is proportional to the total number of forward propagations performed, then the overall computational load for PSO will be in the order $O(SET\mu_t)$. Table 4 presents average training times for three medical benchmark problems by BP and PSO over the sample architecture space given in Table 2. The results clearly indicate that BP is computationally more efficient than PSO with the current parameters of BP and PSO. Since the computational complexity of both methods strictly depends on their parameters (e.g. E for both BP and PSO and S for PSO), it is nevertheless difficult to perform a decisive comparison between them.

4. Conclusions

In this paper, the performance of the two well-known training techniques, BP and PSO, for neural network classifiers over medical

datasets was evaluated by using a new assessment methodology, based on a systematic and exhaustive evaluation of the classifier performance over an architecture space and with respect to different training depths. The proposed assessment method can thus evaluate each method's training (error minimization) and test (generalization) performances with respect to different network configurations and training depths whilst considering two statistical criteria, "on the average" and "the best" within an exhaustive number of (training) runs. An extensive experimental study was performed over the three benchmark medical diagnosis problems from *Proben1* repository. In this study, the architecture space was defined over the feed-forward, fully-connected ANNs (the so-called MLPs), which have been widely used in computer-aided decision support systems in medical domain. The experimental results clearly demonstrate that both training and generalization performance of each method depend on the network configuration, the training depth and the particular application at hand. This is basically what has been missing in the related works in this field and that is why there are many variations and even contradictory or mismatching results among them. We strongly believe that this work has important implications for researchers designing neural network classifiers for medical diagnosis systems.

Regarding the performances of both methods, it is concluded first of all that PSO-training has demonstrated relatively better generalization ability across the architecture space whereas BP with deep training always achieved the lowest MSE levels. Furthermore, the classification performance of PSO shows a strong immunity against variations in the training depth, and it generally achieves the lowest minimum CEs. BP, on the other hand, is consistently better than PSO for compact networks, yet the training performances (minimum and average MSEs) of both BP and PSO are quite comparable and usually varying with different network configurations. Contrary to the results published in Mazurowski et al. (2008), we proved that PSO, in both shallow and deep training schemes, usually yields a better classification performance than BP, especially when the minimum CE is taken into account (i.e. the best classification performance). We have further shown statistically that the performance of BP method is drastically underestimated in Sexton and Dorsey (2000) as it can achieve much better classification performance over the entire architecture space used. Finally, as expected, BP is found to be more computationally efficient than PSO.

The proposed method for performance assessment of ANN classifiers can also be used for other types of ANNs with different architecture spaces and in other application areas. This is subject to our future work, which will focus on the application of the proposed assessment methodology for radial basis function (RBF) neural networks over the same benchmark medical datasets. The results will allow us to compare, not only the training methods, BP vs. PSO, but also both types of ANNs, i.e. MLPs vs. RBFs.

References

- Amari, S., Murata, N., Muller, K. R., Finke, M., & Yang, H. H. (1997). Asymptotic statistical theory of overtraining and cross-validation. *IEEE Transactions on Neural Networks*, 8(5), 985–996.
- Andersen, T., & Martinez, T. (1999). Cross validation and MLP architecture selection. In *Proceedings of international conference on neural networks* (pp. 1614–1619).
- Back, A., & Tsoi, A. C. (1994). On the backpropagation algorithm: Paralysis in multilayer perceptrons. In *Proceedings of the fifth Australian conference on neural networks* (pp. 102–104).
- Back, T., & Kursawe, F. (1995). Evolutionary algorithms for fuzzy logic: A brief overview. In *Fuzzy logic and soft computing* (pp. 3–10). Singapore: World Scientific.
- Baxt, W. G. (1990). Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion. *Neural Computing*, 2, 480–489.
- Carvalho, M., & Ludermir, T. B. (2007). Particle swarm optimization of neural network architectures and weights. In *Proceedings of the 7th international conference on hybrid intelligent systems, Washington DC* (pp. 336–339).

- Chauvin, Y., & Rumelhart, D. E. (1995). *Back propagation: Theory, architectures, and applications*. Associates Publishers, Hillsdale, NJ, USA: Lawrence Erlbaum.
- Christopher, K. M., & Seppi, K. D. (2004). The Kalman swarm. A new approach to particle motion in swarm optimization. In *Proceedings of the genetic and evolutionary computation conference, GECCO* (pp. 140–150).
- Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the IEEE congress on evolutionary computation* (Vol. 3, pp. 1951–1957).
- Eberhart, R. C. & Hu, X. (1999). Human tremor analysis using particle swarm optimization. In *Proceedings of the congress on evolutionary computation* (pp. 1927–1930).
- Fayyad, U. M., Shapire, G. P., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Cambridge, MA: MIT Press.
- Goldberg, D. (1989). *Genetic algorithms in search. Optimization and machine learning*. Reading MA: Addison-Wesley. pp. 1–25.
- Gudise, V. G., & Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE swarm intelligence symposium* (pp. 110–117).
- Haselsteiner, E., & Pfurtscheller, G. (2000). Using time dependent neural networks for eeg classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 8(4), 457–463.
- Haykin, S. (1999). *Neural networks: A comprehensive foundation, 2/E*. New York: Prentice Hall.
- Higashi, H., & Iba, H. (2003). Particle swarm optimization with gaussian mutation. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 72–79).
- Hosseini, H. G., Luo, D., & Reynolds, K. J. (2006). The comparison of different feed forward neural network architectures for ECG signal diagnosis. *Medical Engineering and Physics*, 28, 372–378.
- Hu, X., & Shi, Y. (2004). Recent advances in particle swarm. In *Proceedings of IEEE congress on evolutionary computation* (pp. 90–97).
- Joost, M., & Schiffmann, W. (1998). Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization. *International Journal of Uncertainty, Fuzziness Knowledge-based Systems*, 6(2), 117–126.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks, Perth, Australia* (Vol. 4, pp. 1942–1948).
- Kolen, J. F., & Pollack, J. B. (1990). Back propagation is sensitive to initial conditions. *Complex Systems*, 4, 860–867.
- Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Massachusetts: MIT Press, Cambridge.
- Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*(1987), 4–22.
- Lisboa, P. J. (2002). A review of evidence of health benefit from artificial neural networks in medical intervention. *Neural Networks*, 15, 11–39.
- Lisboa, P. J., & Taktak, A. F. G. (2006). The use of artificial neural networks in decision support in cancer: A systematic review. *Neural Networks*, 19(4), 408–415.
- Lovberg, M. (2002). *Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality*. MSc thesis, Department of Computer Science, University of Aarhus, Denmark.
- Lovberg, M. & Krink, T. (2002). Extending particle swarm optimisers with self-organized criticality. In *Proceedings of the IEEE congress on evolutionary computation* (Vol. 2, pp. 1588–1593).
- Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A., & Tourassi, G. D. (2008). Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21, 427–436.
- Meissner, M., Schmuken, M., & Schneider, G. (2006). Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. *BMC Bioinformatics*, 7, 125.
- Prechelt, L. (1994). *Proben1 – A set of neural network benchmark problems and benchmark rules*. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany.
- Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of the IEEE international conference on neural networks* (pp. 586–591).
- Riget, J., & Vesterstrom, J. S. (2002). *A diversity-guided particle swarm optimizer – The ARPSO*. Technical Report, Department of Computer Science, University of Aarhus.
- Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representation by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 318–362). Cambridge, MA: MIT Press.
- Settles, M., Rodebaugh, B., & Soule, T. (2003). Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In *Lecture notes in computer science (LNCS) no. 2723: proc. of the genetic and evolutionary computation conference 2003 (GECCO 2003), Chicago, IL, USA* (pp. 151–152).
- Sexton, R. S., & Dorsey, R. E. (2000). Reliable classification using neural networks: A genetic algorithm and back propagation comparison. *Decision Support Systems*, 30, 11–22.
- Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 69–73).
- Silverman, R. H., & Noetzel, A. S. (1990). Image processing and pattern recognition in ultrasonograms by backpropagation. *Neural Networks*, 3, 593–603.
- Tourassi, G. D., Floyd, C. E., Sostman, H. D., & Coleman, R. E. (1993). Acute pulmonary embolism: Artificial neural network approach for diagnosis. *Radiology*, 189, 555–558.
- Van den Bergh, F. (2002). *An analysis of particle swarm optimizers*. Ph.D. thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa.
- Van den Bergh, F., & Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimizer. In *Proceedings of the IEEE international conference on systems, man, and cybernetics* (pp. 96–101).
- West, D., & West, V. (2000). Improving diagnostic accuracy using a hierarchical neural network to model decision subtasks. *International Journal Medical Informatics*, 57(1), 41–55.
- Wilson, E. O. (1975). *Sociobiology: The new synthesis*. Cambridge, MA: Belknap Press.
- Yu, J., Xi, L., & Wang, S. (2007). An improved particle swarm optimization for evolving feedforward artificial neural networks. *Neural Processing Letters*, 26(3), 217–231.