Direct Zero-Norm Minimization for Neural Network Pruning and Training

S.P. Adam[1,2], George D. Magoulas[3], and M.N. Vrahatis[1]

[1] Computational Intelligence Laboratory, Dept. of Mathematics, University of Patras, Rion - Patras, Greece
[2] Dept. of Informatics and Telecommunications, Technological Education Institute of Epirus, Arta, Greece
[3] Dept. of Computer Science and Information Systems, Birkbeck College, University of London, United Kingdom

Abstract. Designing a feed-forward neural network with optimal topology in terms of complexity (hidden layer nodes and connections between nodes) and training performance has been a matter of considerable concern since the very beginning of neural networks research. Typically, this issue is dealt with by pruning a fully interconnected network with "many" nodes in the hidden layers, eliminating "superfluous" connections and nodes. However the problem has not been solved yet and it seems to be even more relevant today in the context of deep learning networks. In this paper we present a method of direct zero-norm minimization for pruning while training a Multi Layer Perceptron. The method employs a cooperative scheme using two swarms of particles and its purpose is to minimize an aggregate function corresponding to the total risk functional. Our discussion highlights relevant computational and methodological issues of the approach that are not apparent and well defined in the literature.

Keywords: Neural networks, pruning, training, zero-norm minimization, Particle Swarm Optimization

1 Introduction

Despite the popularity of neural network models in engineering applications designing a neural network for a particular application remains a challenge. For example, in system identification using neural networks, finding the optimal network architecture is not straightforward [1]. In addition designing a neural network for a control system is critical for optimal performance [2]. Besides the type of the nodes' activation functions and network training parameters this design also involves selecting the right number of hidden layer nodes and their interconnection. Thus, practitioners and researchers invested in pruning techniques for defining the best available neural network architecture [1], [2].

The question on the number of hidden layers and more specifically on the number of nodes per hidden layer necessary to approximate any given function $f : \mathbb{R}^n \to \mathbb{R}^m$, has been tackled by several researchers, as noted in [3], and has

C. Jayne, S. Yue, and L. Iliadis (Eds.): EANN 2012, CCIS 311, pp. 295–304, 2012.

been studied as part of the problem of the density of Multi Layer Perceptrons (MLPs). Furthermore, work introduced in [4], [5], [6] and [7] highlighted important theoretical aspects of the density problem and derived bounds on the number of nodes of the hidden layer for one hidden layer MLPs. However, as noted in [6], "in applications, functions of hundreds of variables are approximated sufficiently well by neural networks with only moderately many hidden units". No precise rules exist regarding the necessary and sufficient architecture of the neural network to deal with some specific problem. It is common when designing a network to apply a pruning technique. This consists in designing a "fat" network with many nodes in the hidden layer and fully interconnected layers. Then proceeding with detecting those connections and/or nodes that are superfluous, with respect to the mapping function of the network, and removing them. A number of pruning techniques and related research are reported in [8].

The approach proposed in this paper is a network pruning technique. Section 2 is devoted to the problem background and a brief literature review. In section 3 the problem of zero-norm minimization is formulated. Section 4 discusses how cooperative swarms are used in this minimization problem and section 5 presents application experiments. The paper ends with some concluding remarks.

2 Background and Previous Work

The whole problem is defined as the complexity-regularization problem and falls within Tikhonov's regularization theory, [9]. From that point of view, defining the architecture of a network and training it in order to maximize generalization performance in the context of supervised learning, is equivalent to minimizing the total risk given by the following expression, as noted in [10],

$$\mathcal{R}(w) = \mathcal{E}_S(w) + \lambda \mathcal{E}_C(w) \ , \tag{1}$$

where $\mathcal{E}_S(w)$ is the standard performance measure, typically the error function of the network's output, $\mathcal{E}_C(w)$ is a penalty term, a functional that depends exclusively on the network architecture, that is the network's complexity, and λ is a real number whose value determines the importance attached to the complexity penalty term when minimizing the total risk.

The so called penalty term methods are derived from the formulation of (1). These methods tend to minimize the total risk functional by defining weight decay or weight elimination procedures which are applied during network training with back-propagation and hunt out "non-significant" weights by trying to drive their values down to zero. This means that the corresponding connections are eliminated and in consequence nodes with no connections at all are also eliminated. Typical methods in this category are proposed in [11] and [12].

Another group of methods tend to eliminate connections and/or nodes by calculating various sensitivity measures such as the sensitivity of the standard error function to the removal of units (Skeletonization, [13]), the removal of connections [14] and LeCun's Optimal Brain Damage, [15]. In general, such methods act on the network architecture and modify it once the network is trained. Among

several other methods that have been proposed it is worth mentioning those that are hessian-based, [16], or variance-based pruning techniques, see [8].

A number of methods have also been defined based on Evolutionary Computation. Hancock in [17] analyzes the aspects of pruning neural networks using the Genetic Algorithm (GA). Methods reported in the literature use the GA for network architecture optimization as well as for weight training. However, as noted in [18], GA may not yield a good approach to optimizing neural network weights because of the Competing Convention Problem, also called the Permutations Problem. Some research efforts propose hybrid approaches where the network training task is carried out by backpropagation.

In particular, it is worth mentioning here that some recent pruning techniques adopt the Particle Swarm Optimization (PSO) paradigm. One approach is proposed in [19] and uses a modified version of PSO in order to determine the network architecture (nodes and connections) as well as the activation function of the nodes and the best weight values for the synapses. Results of the experiments presented give relatively complex networks having intra-layer connections, direct input-to-output connections and nodes of the same layer with different activation functions. The subsequent question deals with how easily these networks can be implemented for real life applications. Another approach is presented in [20] and uses a so-called cooperative binary-real PSO to tune the structure and the parameters of a neural network. Researchers' main assumption is that the signal flow via the connections of the nodes is controlled by ON/OFF switches. This engineering consideration is implemented using a binary swarm together with a real valued swarm which is used to train the network. Despite the results reported, our opinion is that, there are two points that remain unclear in this paper. The first deals with the exact mode of interaction between the two swarms. The second concerns the problem of state space exploration with particles traveling around in different subspaces of the state space. The latter will be discussed and clarified later in section 4. Lastly, some hybrid approaches can be found in the literature using both PSO and classical backpropagation for weight training.

The approach presented in this paper is a penalty term method for neural network regularization. The penalty term is based on the zero-norm of the vector formed by the weights of the network. In contrast to [20] our approach has a clearly defined model of interaction between the two swarms and an effective mechanism for state space exploration. Lastly, our work is underpinned by a mathematical theory for dealing with the neural network pruning problem.

3 Minimizing the Zero-Norm of Weights

Let $\bar{w}$ denote the vector formed by the weights of the network connections arranged in some order. Then the zero-norm of this vector is the number of non-zero components, also defined by the expression, $\|\bar{w}\|_0 = card\{w_j,\ w_j \neq 0\}$. Given that, pruning a network is eliminating connections, or in other words zeroing corresponding weights, then pruning a network can be considered as minimizing

the zero-norm of the weight vector of the network. Use of the zero-norm has been proposed, in the machine learning literature, as a measure of the presence of some features or free parameters in learning problems when sparse learning machines are considered. The reason for using zero-norm minimization in machine learning is that, seemingly, it provides a natural way of directly addressing two objectives, feature selection and pattern classification, in just a single optimization, [21]. In fact, these two objectives are directly interconnected, especially regarding regularization which enforces sparsity of the weight vector. One may easily notice that pruning a network can, also, be considered as a feature selection problem, and, thus, the above considerations, again, justify its rephrasing as a zero-norm minimization problem.

Although zero-norm minimization has been used by many researchers in various machine learning contexts; [22], it is important to point out an important theoretical issue in the context of neural networks that is computational complexity when minimizing the zero-norm of some vector. Amaldi and Kann in [23], proved that minimizing the zero norm of some vector, let $\|\bar{w}\|_0$, subject to the linear problem $y_i\,(wx_i + b) \geq 1$ is a problem that is NP-complete. In practice researchers address zero-norm minimization using some approximate form such as, $\sum_i \left(1 - e^{-\alpha|w_i|}\right)$, where α is a parameter to be chosen, or $\sum_i \log_{10}(\varepsilon + |w_i|)$, where $0 < \varepsilon \ll 1$.

Driving weights to zero or deleting weight in order to eliminate connections between nodes has been fundamental in a number of network regularization approaches. Such pruning techniques are weight decay procedures which are applied combined with gradient descent-based training methods. These weight decay procedures use a differentiable form which is based on some norm of the weight vector. Typical examples are the methods proposed by Hinton, [11], and Moody and Rögnvaldsson, [24]. It is worth noting here that Rumelhart, as reported in [25], investigated a number of different penalty terms using approximate forms of some weight vector norms. In particular, the penalty term $\sum_i \left[\left(w_i^2/w_0^2\right) / \left(1 + w_i^2/w_0^2\right)\right]$ proposed by Weigend et al. [12], constitutes an approximate form of the zero-norm of the weight vector.

Our approach to network pruning adopts direct minimization of the zero-norm of the weight vector. Hence, the values of selected weights are zeroed without decay (directly) and thus corresponding node connections are cutoff abruptly. Such a consideration gives rise to the connectivity pattern of a network, i.e. a binary vector where each component denotes whether a connection is present, with 1, or 0 if it is deleted. This vector can be considered as the connectivity adjoint of the weight vector. Thus, using an indicator function the weight vector $\bar{w} = (w_1, w_2, ..., w_n)$, is replaced by the binary vector defined as $\mathbb{1}_{\bar{w}} = (\mathbb{1}_{w_1}, \mathbb{1}_{w_2}, ..., \mathbb{1}_{w_n})$, where $\mathbb{1}_{w_i} = \begin{cases} 1 & w_i \neq 0 \\ 0 & w_i = 0 \end{cases}$, $1 \leq i \leq n$. In consequence, the penalty term of Equation (1) becomes $\mathcal{E}_C(w) = \lambda \|w\|_0$ or using the connectivity vector representation $\mathcal{E}_C(w) = \lambda \sum_{i=1}^{n} \mathbb{1}_{w_i}$. Finally, if the mean squared error

of the network output is used as the standard measure of the network performance the aggregate function (1) takes the form,

$$\mathcal{R}(w) = \frac{1}{p}\frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n}\left(d_i^k - o_i^k\right)^2 + \lambda\sum_{i=1}^{n}\mathbb{1}_{w_i} \ . \tag{2}$$

This form of the penalty term is not differentiable. So its minimization cannot be based on classical back-propagation and an evolutionary approach should be adopted. Minimizing an aggregate function of the form (2) has been addressed as a multi-objective optimization problem and several methods have been proposed in the context of evolutionary computation. Our approach is based on PSO and specifically on a cooperative scheme that uses two swarms. This scheme is relative to the approaches developed in [26] and [27].

4 Implementation of Cooperative Swarms

The cooperative scheme adopted for the minimization of this aggregate function makes use of two swarms; let them be TrnSrm (Training Swarm) and PrnSrm (Pruning Swarm). TrnSrm undertakes the minimization of the term corresponding to the standard error of the neural network output and so it performs network training. On the other hand, PrnSrm aims at minimizing the zero-norm of the network's connectivity vector, thus implementing the pruning function. It is obvious that these two terms of the risk functional and therefore the two swarms are very strongly coupled. During optimization we expect PrnSrm to derive the thinnest possible network architecture that performs well in terms of training and generalization that is permitting TrnSrm to reach some "good" local minimum for the network error function.

MLPs considered here have only one hidden layer and are fully interconnected without lateral or backward connections between the nodes. The assumption regarding the number of layers is not restrictive for the proposed method as none of the statements, herein, depends on this specific hypothesis. Let N be the number of weights of such a neural network. Each particle in TrnSrm corresponds to a network that constitutes a possible solution to the standard error term of Equation (2). Such a particle is a vector in $\mathbb{R}^N$ whose values correspond to the network weights. Particles in PrnSrm are binary. Each particle describes a possible network configuration represented by a binary vector in $\{0,1\}^{N-B}$, where B is the number of bias connections. Hence, only connections between nodes are used to form the binary particles. There is a one-to-one correspondence between particles in TrnSrm and particles in PrnSrm. Thus, any change in a particle in PrnSrm implies a change in the architecture of the corresponding particle in TrnSrm. The fitness function for TrnSrm is either the mean squared or the mean absolute error of the network output, while the fitness function for PrnSrm is $\sum_{i=1}^{N-B}\mathbb{1}_{w_i}$.

A methodological issue when defining and minimizing the aggregate form (1) is the choice of λ. If the form $\mathcal{R}(w) = \lambda_1\mathcal{E}_S(w) + \lambda_2\mathcal{E}_C(w)$, where $\lambda_1 + \lambda_2 = 1$ is

used instead of (1) then the relation between λ_1 and λ_2 determines the priority given by the minimization process to each of the two terms. In this context, promoting some network architecture, detected by PrnSrm, depends on the estimation that this architecture is likely to lead or even be an optimal solution for the minimization problem. The measure of this estimation is λ_2. In our approach defining λ_1 and λ_2 relies on the following considerations.

The overall minimization task is carried out by two populations which, though tightly coupled, operate separately. However, when the pruning swarm is left to operate alone, observation shows that about 80% of the components of a binary vector are zeroed in about 10% of the total number of iterations, Fig. 1. During this number of iterations for a simple problem such as the Iris classification benchmark, using a 4-5-3 network, the training swarm operating alone achieves to reduce the network output error by about 40% of the initial value, Fig. 1. This rough observation underlines the consideration that, for every network architecture offered by PrnSrm as a solution, TrnSrm should be given the time (number of iterations) to verify that the proposed architecture can be trained to minimize the standard error function. Furthermore some estimation should be made about the reliability of this result. This suggests setting $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$.

Effective implementation of the proposed approach is equivalent to having

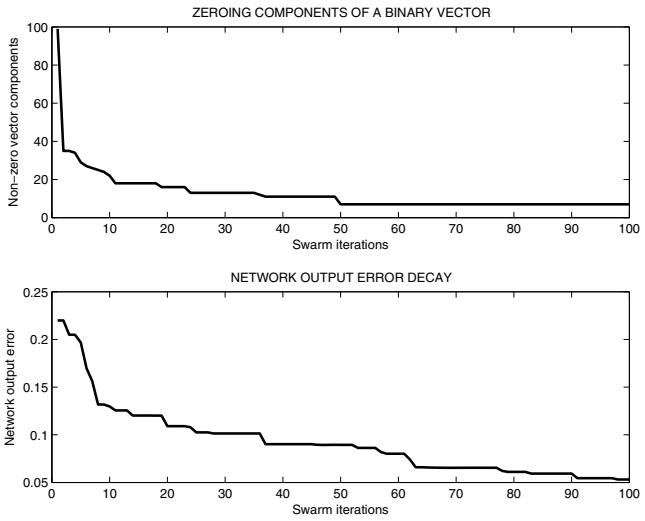

Fig. 1. Indicative minimization rate for the two swarms

a computing scheme consisting of two separate processes operating in parallel that need to be synchronized in order to achieve the optimal solution in terms of network complexity and performance. At any time the aggregate function reflects the problem's best state. The training swarm operates in order to reduce

the output error of the best network architecture to some significant level. For our experiments this level is set to 70% of the output error computed for the best network architecture selected every time the training swarm is launched. If this goal is reached then the pruning swarm is activated to further reduce the number of connections of the network architecture contributing the aggregate function. Once such a new architecture is found the training swarm is launched again.

Another important issue for the exploration process concerns the order in which connections are eliminated. For instance suppose that there are K particles in each swarm. Following an iteration of PrnSrm these K particles suggest k possible network architectures, where $k \leq K$. Considering that, when a connection is eliminated the corresponding dimension in the N-dimensional weight space is not explored by the optimization process, it is easy to realize that a random cutoff of network connections disperses the particles around in different subspaces of the N-dimensional weight space. Thus the exploration process weakens in the sense that a global best may become non-significant for particles exploring different subspaces. The obvious consequence is that the optimization process may fail to converge. The solution we adopted for this problem is to apply connection cutoff in some ordered way by progressively eliminating input connections and output connections of successive nodes of the hidden layer. This is a kind of "normalized connection elimination" which ensures that at any time there exists a subspace of the weight space that is common to and explored by all particles in the swarm.

5 Experimental Evaluation

In order to validate the proposed approach we carried out a number of experiments on four well known real world problems, the Fisher Iris classification benchmark, the Servo prediction, the Solar sunspot prediction problem and the Wine classification problem. All data sets are from the UCI repository of machine learning database. Note that two alternative approaches were used for the Solar sunspot prediction problem. The first (Solar1) is the most typical one found in the literature, while the second (Solar2) is used for comparison with the approach presented in [20]. Table 1, summarizes typical network architectures found in the literature and structures used for the experiments.

The experimental setup for the proposed approach comprises two swarms as described above, each one consisting of 50 particles. The classic algorithm, [27], without enhancements is used for updating the velocities of the particles. The max number of iterations for the swarms is set to 1000 and values for the cognitive and the social attraction parameters are set to 0.5 and 1.25 respectively. For each network used, a set of 100 instances were derived and pruning experiments were executed using the proposed approach and two classical pruning methods, the Optimal Brain Damage (OBD) [15] and the Optimal Brain Surgeon (OBS), [16], as implemented in NNSYSID20, [1].

Table 1. Number of nodes and connections for networks used in experiments

	Typical network architectures		Network architectures used	
	Nodes	Connections	Nodes	Connections
Iris	4-2-3	14 (+5 bias)	4-20-3	140 (+23 bias)
			4-10-3	70 (+13 bias)
Servo	12-3-1	39 (+4 bias)	12-8-1	104 (+9 bias)
Solar1	12-5-1	65 (+6 bias)	12-15-1	195 (+16 bias)
Wine	13-6-3	96 (+9 bias)	13-15-3	240 (+18 bias)
Solar2	3-3-1	12 (+4 bias)	3-8-1	32 (+9 bias)
			3-15-1	60 (+16 bias)

Table 2, hereafter, summarizes the results obtained for the following parameters: Nodes pruned (Npr) in hidden layer, Connections pruned (Cpr) and Generalization achieved (Gen). Values reported for generalization for the prediction problems are mean value and standard deviation. Generalization performance is the mean value of the percentage of correctly classified test patterns for the Iris and the Wine problems. For the Servo and the Solar sunspot prediction problems generalization is the mean absolute error between the network output and the expected output of the test patterns.

Table 2. Results of the experiments

	Proposed Method			OBS			OBD		
	Npr	Cpr	Gen	Npr	Cpr	Gen	Npr	Cpr	Gen
Iris	12	86	95.56%	0.6	0	34%	0	0	34%
	6	45	97.78%	2	41	86.67%	6	40	68.89%
Servo	5	65	0.0515 0.0091	1	42	0.0721 0.0309	1	43	0.0716 0.0329
Solar1	9	114	0.0905 0.0092	0	1	0.9094 0.2468	0	1	0.9094 0.2468
Wine	9	139	97.78%	0	0	34%	0	0	34%
Solar2	6	23	0.0770 0.0050	13	3	0.0693 0.0085	2	10	0.2190 0.3000
	10	39	0.0765 0.0058	8	28	0.0693 0.0164	24	11	0.2190 0.3925

It is worth noting that due to the way connections are eliminated the proposed method does not provide sparse networks in terms of having few connections between dispersed nodes in the hidden layer. One may notice that the two classic methods OBS and OBD fail when faced with really "fat" networks, while our approach achieves to eliminate "fat" from the network even in difficult situations. When compared against the relative method presented in [20] (experiments under the Solar2 label) the proposed method demonstrates clearly better performance in terms of pruning, while generalization cannot be compared as the authors in [20] do not provide enough data. Finally we need to note that despite the small number of iterations (1000) for the swarms the method converged

in more that 80% of the cases. This score overrides the defect reported in the literature that classic weight decay methods fail to converge, [25], and so they are not used in practice.

6 Concluding Remarks

The proposed methods introduces zero-norm minimization for pruning the weights of an MLP. Besides the definition of a solid mathematical basis the method clarifies two important implementation points which are not clear with other methods [19], [20] and greatly affect convergence of the minimization process. Experiments carried out and results obtained, in Table 2 above, clearly reveal the ability of the proposed method to eliminate connections and nodes while training the network. Networks obtained after pruning are considered optimal in the sense that the number of nodes and connections are very close to the values typically found in the literature for the benchmarks under consideration. When compared against two classic pruning methods the proposed approach performs better both in terms of pruning and training, and thus it breaks the non-convergence deficiency associated with typical weight decay methods. However, in terms of training the approach can be further improved considering that the ability of classic PSO algorithm is relatively poor regarding local search. In this paper we did not elaborate this issue which will be further investigated in future work.

References

1. Norgaard, M.: Neural Network Based System Identification Toolbox, version 2. Technical report, 00-E-891, Dept. of Automation, Technical University of Denmark (2000)
2. Stepniewski, S.W., Keane, A.J.: Topology Design of Feedforward Neural Networks by Genetic Algorithms. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 771–780. Springer, Heidelberg (1996)
3. Pinkus, A.: Approximation theory of the MLP model in neural model. Acta Numerica, 143–195 (1999)
4. Jones, L.K.: A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. The Annals of Statistics 20, 601–613 (1992)
5. Barron, A.R.: Universal approximation bounds for superposition of a sigmoidal function. IEEE Trans. Inform. Theory 39, 930–945 (1993)
6. Kůrková, V., Kainen, P.C., Kreinovich, V.: Estimates of the number of hidden units and variation with respect to half-spaces. Neural Networks 10, 1061–1068 (1997)
7. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks 4, 251–257 (1991)
8. Reed, R.: Pruning algorithms - A Survey. IEEE Trans. Neural Networks 4, 740–747 (1993)
9. Tikhonov, A.N., Arsenin, V.Y.: Solution of Ill-posed Problems. W.H. Winston, Washington, DC (1977)

10. Haykin, S.: Neural networks: A comprehensive Foundation. Prentice-Hall, Upper Saddle River (1999)
11. Hinton, G.E.: Connectionist learning procedures. Artificial Intelligence 40, 185–234 (1989)
12. Weigend, A.S., Rumelhart, D.E., Huberman, B.A.: Generalization by weight-elimination with application to forecasting. In: Lippmann, R., Moody, J., Touretzky, D. (eds.) Advances in Neural Information Processing Systems (3), pp. 875–882. Morgan-Kaufmann, San Mateo (1991)
13. Mozer, M.C., Smolensky, P.: Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems (1), pp. 40–48. Morgan Kaufmann, San Francisco (1989)
14. Karnin, E.D.: A simple procedure for pruning back-propagation trained neural networks. IEEE Trans. Neural Networks 1, 239–242 (1990)
15. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal Brain Damage. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems (2), pp. 598–605. Morgan Kaufmann, San Francisco (1990)
16. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal Brain Surgeon. In: Hanson, S.J., Cowan, J.D., Giles, C.L. (eds.) Advances in Neural Information Processing Systems (5), pp. 164–172. Morgan-Kaufmann, San Mateo (1993)
17. Hancock, P.J.B.: Pruning neural networks by genetic algorithm. In: Aleksander, I., Taylor, J.G. (eds.) Proc. of the International Conference on Artificial Neural Networks, pp. 991–994. Elsevier, Brighton (1992)
18. Whitley, D.: Genetic Algorithms and Neural Networks. Genetic Algorithms in Engineering and Computer Science, pp. 191–201. John Wiley (1995)
19. Garro, B.A., Sossa, H., Vazquez, R.A.: Design of artificial neural networks using a modified particle swarm optimization algorithm. In: Proc. IEEE International Joint Conference on Neural Networks, Atlanta, pp. 938–945 (2009)
20. Zhao, L., Qian, F.: Tuning the structure and parameters of a neural network using cooperative binary-real particle swarm optimization. Expert Systems with Applications (2010)
21. Weston, J., Elisseeff, A., Schölkopf, B., Tipping, M.: Use of the zero-norm with linear models and kernel methods. J. Machine Learning Res. 3, 1439–1461 (2003)
22. Fung, G.M., Mangasarian, O.L., Smola, A.J.: Minimal kernel classifiers. J. Machine Learning Res. 3, 303–321 (2002)
23. Amaldi, E., Kann, V.: On the approximability of minimizing non zero variables or unsatisfied relations in linear systems. Theoretical Computer Science, 237–260 (1998)
24. Moody, J.E., Rögnvaldsson, T.: Smoothing regularizers for projective basis function networks. In: Mozer, M., Jordan, M.I., Petsche, T. (eds.) Advances in Neural Information Processing Systems (9), pp. 585–591. MIT Press, Denver (1997)
25. Hanson, S.J., Pratt, L.Y.: Comparing biases for minimal network construction with back-propagation. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems (1), pp. 177–185. Morgan Kaufmann, San Francisco (1989)
26. Parsopoulos, K.E., Tasoulis, D.K., Vrahatis, M.N.: Multi-objective optimization using parallel vector evaluated particle swarm optimization. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications (AIA), Innsbruck, vol. 2, pp. 823–828 (2004)
27. van de Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evolutionary Computation 8, 1–15 (2004)