https://doi.org/10.26552/com.C.2000.2.38-55

komunikácie
COMMUNICATIONS

Peter Géczy - Shiro Usui *

INTELIGENTNÉ ADAPTABILNÉ SYSTEMY: PRÍSTUP PRVÉHO RÁDU

INTELLIGENT ADAPTABLE SYSTEMS: FIRST ORDER APPROACH

Budúcnosť komunikácií nevyhnutne vyžaduje technológie umožňujúce vysokú úroveň flexibility, adaptability a inteligencie. Inteligentné adaptabilné systémy sú obzvlášť vhodné pre túto úlohu. Väčšina adaptabilných systémov je založená na neurónových sieťach. Umelé neurónové siete sú systémy s enormnou interkonektivitou. Neurónové siete sa neprogramujú. Sú schopné nadobudnúť významné vlastnosti vďaka adaptačnému procesu nazývanému učenie. Predkladaný inteligentný adaptabilný systém využíva technológiu neurónových sietí. Systém umožňuje internú viacúrovňovú adaptabilitu. Vzhľadom na dostupné dáta autonómne adaptuje svoje parametre a štruktúru. Z externej perspektívy disponuje kontrolou vlastného vstupno-výstupného interfejsu. Systém je schopný vybrať si vhodné učiace exempláre z dostupného množstva dát tak, aby dosahoval optimálny progres učenia. Po naučení systém disponuje možnosťou výstupu v logickom formáte. Uvedený inteligentný adaptabilný systém pozostáva z niekoľkých modulov. Princíp a funkcia každého z modulov sú popísané a ilustratívne demonštrované.

Future of communications inevitably calls for technologies that feature high-level flexibility, adaptability, and intelligence. Intelligent adaptable systems are particularly suitable for this mission. Majority of adaptable systems utilize neural networks. Artificial neural networks are systems with huge network-like interconnectivity. They are not programmed. Neural Networks gain valuable properties through the process of adaptation called learning. Presented intelligent adaptable system utilizes neural network technology. The system incorporates internal adaptability at several levels. It autonomously adapts its parameters and structure to the presented data. Externally, it appropriately manages its input-output interfaces. The system is able to select suitable training exemplars from the available amount of data in order to achieve the optimal learning performance. After training the system provides logical output format of the task. Introduced intelligent adaptable system consists of several modules. Principle and functionality of each module is described and illustratively demonstrated.

1. Introduction

Rapid expansion of communication technologies is widely influencing our everyday life. Userbase of communication services grows at such a rate that technology development constantly faces challenges in order to sustain stability and reliability of services. Such state of the communications sector has led to the need of developing new techniques to enhance speed, expand bandwidth, and provide higher security, to mention a few.

Neural networks, as a result of their inherent learning and adaptive qualities, have enormous applicability in this explosive area of technology. Articial neural networks represent the technical abstraction of the biological structures observable in the nervous system of living creatures. The nervous system of biological entities consists of the elementary processing blocks-neurons. Neurons are interconnected by synapses and axons, enabling propagation of bio-signals and creation of bio-information pathways. Huge interconnectivity allows formation of wide networks with massive parallel processing capabilities. Just as neural networks, the com-munication systems are composed of units that process signal/information transmitted over cables or ether using well-defined protocols such as TCP/IP, ISDN, CDMA, TDMA, etc. This apparent parallel uncovers enormous potential of neural networks in communication technologies of future. It positions the neural networks into a place where a considerable advantage of their adaptability (and other properties) can be taken.

At present the communication technologies could be viewed as algorithm-oriented. Switching, routing, and packet transmitting is controlled by algorithms that feature none or very little adaptability. This trend has originated from the classical information science established in the early 50's by Shannon [1] and has further been strengthened by development of computer science. The nodes in communication infrastructure, though they are highly so-phisticated, automated, and computerized systems, utilize the algorithmic concepts lacking higher-order auto-adaptability to external and/or internal conditions. Lack of adaptability results in lower effectiveness of the global communication infrastructure as well as its elementary units.

The future of communication technologies inevitably calls for higher adaptability and/or learning, flexibility, and "intelligence". With the current advancement of neural networks all these qualities can be achieved using conventional computer systems without substantial investment in rebuilding the existing physical communication infrastructure. This economical solution is likely to determine the future course of communication technologies.

* **Peter Géczy, Shiro Usui**
Future Technology Research Center, Toyohashi University of Technology, Hibarigaoka, Tempaku-cho, Toyohashi 441-8580, Japan

Novel technologies will impact the communication sector in two major spheres: global and local. Globally influential future communication technologies are required in areas such as network management & control, resource allocation, market prediction, security, reliability, fault tolerance, and datamining. Intelligent adaptable systems (IAS) can be applied directly (or indirectly) to all these domains. In the local domain, IAS have already been applied to routing algorithms (classical traveling salesman problem), voice, image & character recognition, intelligent search & information filtering, and access control [2]. However, the already wide spectrum of IAS applicability is only the initial stage. Further expansion and progress of communication technologies and services will uncover numerous other areas where IAS will be the preferred technological choice.

2. Elements of Intelligent Adaptable Systems

Intelligent adaptable systems draw on the parallel of brain-style information processing. All the changes in the adaptivity of the brain as well as stimuli selectivity occur simultaneously. At the same time the brain is capable of processing incoming information from the receptors, producing adequate responses, and yet adapting itself synaptically and structurally. Although the processing speed of neurons is relatively slow (in the range of milliseconds) the massive parallelism in the brain sufficiently subsidizes this ineficiency. Due to the massive parallelism, the brain is continuously able to process an enormous amount of information.

Block structure of brain-like IAS is depicted in Fig. 1. The system manages its input interface by selecting appropriately the most suitable exemplars for learning. The central part of the system is an artificial neural network. The artificial neural network consists of elementary computational units-artificial neurons-interconnected by real valued weight connections. The artificial neural network should be adaptable both at the microstructure level (connection adaptation) and at the macrostructure level (structural adaptation). The concept displayed in Fig. 1, however, does not end only with the dynamic adaptability and exemplar selectivity. It represents a progressive step forward-towards the logical representation of knowledge that a network gains by training. This is the task of the knowledge acquisition module. The knowledge acquisition module should be able to extract knowledge that the network acquires during the learning. It is also important to note that the subsystems should dynamically co-operate and simultaneously operate on the artificial neural network during the process of learning in order to achieve satisfactory performance on the trained task.

Sample selection is the task of selecting suitable training exemplars from a training data set with the aim of improving the quality of training. Early approaches to sample selection have focused only on determining the informatively sucient subset of training set which was thereafter fixed and used for training. An informatively sufficient sample set was determined based on either the worst case analysis (VC dimensions) or the average case analysis (Bayesian statistics, information theory) [3]. More recent approaches, particularly those linked with on-line training, have

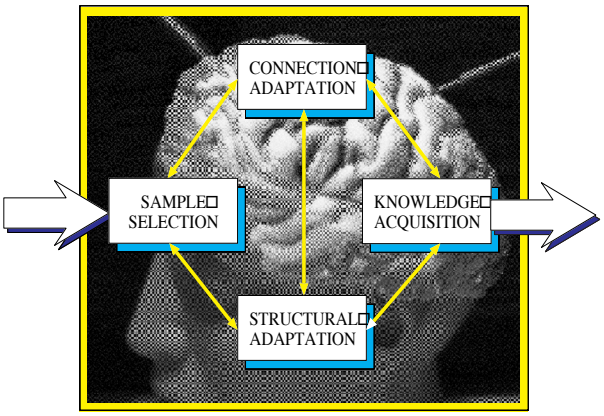

Fig. 1 Block scheme of brain-like IAS. Each module represents a specific underlying feature of learning. A knowledge acquisition module enables the extraction of knowledge from an artificial neural network trained on a specific task in the form of logical rules. All the modules should dynamically and simultaneously co-operate.

focused on determining the sample distribution with respect to which they actively sample data [4] – [8].

None of the aforementioned approaches are used in this study. The dynamic sample selection technique, presented in this study, does not impose any restrictions on the sample distribution and neither on the selected sample size. The number of the selected exemplars is allowed to vary at each iteration. Also the particular training set selected at one iteration may completely differ from the one selected at another iteration. The artificial neural network is given the freedom to select the exemplar set suitable for the fastest progress at each iteration.

The presented dynamic sample selection is capable of selecting an appropriate exemplar set that may vary in size and in the selected samples at each iteration of training. It is based on the controlled normed expression of error gradient that largely determines the search direction of the optimization technique. The error gradient is formed of gradients for each sample presented to a network. Thus the selection of training exemplars also plays a role in determining the search direction of an optimization technique. An appropriately determined search direction formed of suitably selected exemplars at each iteration may increase the convergence speed of optimization and hence also training.

Conection adaptation in neural networks is seen as an optimization task. The objective is to minimize the discrepancy between a network mapping and some true mapping with respect to the measure denoted as the objective function. True mapping is usually not available in its entirety. Having complete information about the true mapping would make the network's training meaningless unless a benchmark evaluation is under consideration. However, even in such cases it is preferred to use benchmark data sets of well-known real-world problems. True mapping is thus represented in its incomplete form, that is, in a form of a finite number of samples.

Relevance of the optimization field [9] in neural network training gives rise to the use of several optimization methods. In

this study preference is given to first order line search optimization techniques [10] – [19], due to their relative computational inexpensiveness, yet reasonable convergence speed. Second order optimization techniques may reach faster convergence rates at the expense of a larger number of calculations. They usually require second order information, that is, the Hessian matrix or its approximation [20] – [23]. Calculation of the Hessian matrix or its approximation becomes pointless for large data sets and/or large network structures due to unbearable computational expensiveness and memory requirements. The fastest achievable convergence rates of first order line search techniques are superlinear convergence rates. Computational complexities of first order approaches are linear, which is one order lower than that of second order techniques. First order approaches have also lower memory requirements. To be specific, the steepest descent methods have no memory requirements and conjugate gradient techniques have linear memory requirements.

Structural adaptation in neural networks aims at optimizing the structure and yet resulting in a network with the required properties. Structural optimality may have a positive impact on generalization property of neural networks. Non-optimality of the structure may cause several complications during and also after training. Underdetermined network structures are incapable of satisfactorily performing the task given by the training set. Overdetermined networks normally display unwanted overfitting properties after training. These and other complications can be avoided by properly fitting the structure of a neural network as well as its parameters.

Former approaches to the structural adaptation of neural networks were based either on regularization or smoothing. The application of regularization techniques results in the modification of the objective function with respect to which the network's parameters are adapted [24], [25]. The objective function then contains a penalizing term for weight connections. The penalizing term leads to connection value-spread. Some connections are forced to take higher values while others lower. Connections having low values are classified as irrelevant and thus eliminated from a structure. Connections with low values are statically irrelevant for mapping performed by a neural network, however, they may have a dynamic relevance during training which may help the network to progress faster. Smoothing approaches are essentially nondynamic [26] – [28]. The network undergoes structural adaptation after training. Curvature of the error surface is evaluated after training and the structural elements of a network corresponding to low curvature values are then removed.

The approach to structural adaptation presented in this study is based on dynamic performance measures [29] – [31]. Performance measures monitor the combined dynamic and static performance of a network up to its particular structural elements. They also detect disturbing features of error surface. Controlling the performance of a network by eliminating the low-performance structural elements and increasing the performance of a network by adding new structural elements represents a new concept for dynamic structural adaptation of neural networks.

Knowledge acquisition delineates a step towards logical representation of a network's "knowledge" gained by training. Since artificial neural networks outline an abstraction of brain structures, it has been a concern of researchers to identify how a network's gained knowledge can be extracted. Distributed representation of knowledge in neural networks makes it a difficult task. Artificial neural network structures do not correspond to any logical representation of knowledge known to humans. A particularly suitable representation of knowledge would be that of logical rules. Such an achievement may have wide impact not only on the design of a new generation of knowledge based systems, but also on our understanding of knowledge representation in the brain.

Currently available approaches to knowledge extraction from artificial neural networks can be viewed from the perspective of rule types and/or training approaches to neural networks predetermined to rule extraction [32] – [35]. The type of rules leads to the distinction of crisp from fuzzy rule extraction techniques. Training approach mainly separates the rule extraction techniques using structure adaptable training and static training. All of the aforementioned approaches, however, strongly precondition the training or the structure of a neural network for further rule extraction tasks. This means, for example, that a neural network may contain nodes representing logical functions or fuzzy membership functions, the structure of the network may be predetermined according to the intended rules to be extracted, a priori knowledge about the task can be implemented into artificial neural networks, etc. All of these techniques reduce the complexity of the principal problem of rule extraction from artificial neural networks.

This study approaches the rule extraction issue in its principal form, that is, independent of the training strategy for neural networks and without preconditioning either the network's structure or processing elements. It allows the neural network to learn freely the task given by the training set. Once the network achieved satisfactory performance in that it maps training data sufficiently correctly, the rule extraction method is applied to transform the network's knowledge into the form of logical rules. The rule extraction method is derived only on the basis of a network mapping and is independent of connection adaptation, sample selection, and structural adaptation [36] – [39].

3. Foundations of the Approach and Notation

Multilayer artificial neural networks, or multilayer perceptrons, are the primary interest in this study. Essentially, a multilayer network is a network which has one input layer, one output layer, and one or more hidden layers. The number of hidden layers can theoretically be unlimited. However, practically they do not exceed several layers. Although, for some application oriented purposes, one can find multilayer perceptron networks with more than one hidden layer, theoretically it has been proven that only one hidden layer is sufficient for universal approximation capabilities of neural networks [40], [41]. This means that an arbitrary functional dependency can be approximated to an arbitrary level of accuracy by a three-layer artificial neural network with an appropriate number of hidden units. Hence the focus on three-layer perceptron networks with the following structure (see Fig. 2).

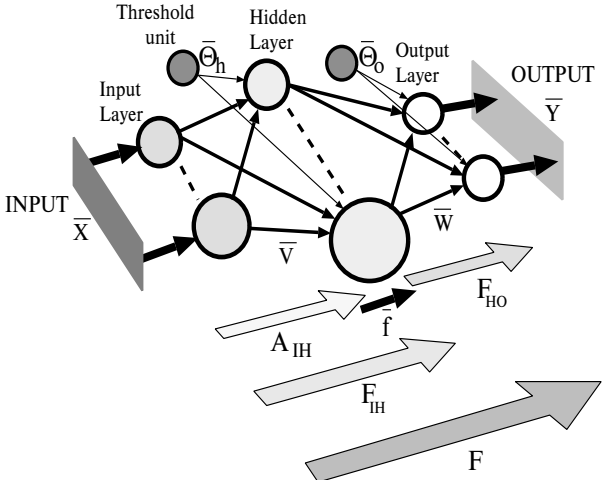

Fig. 2 A model of a three-layer artificial neural network with a mapping scheme. The network contains input, hidden, and output layers. The input layer distributes input signals into the hidden units. The hidden units are nonlinear elements with sigmoidal nonlinear transfer functions. The output units have linear transfer functions. The network's mapping F is decomposed into the input-to-hidden submapping F_{IH} and the hidden-to-output submapping F_{HO} ($F = F_{HO} \circ F_{IH}$). The input-to-hidden submapping F_{IH} ($F_{IH} = \bar{f} \circ A_{IH}$) is further decomposed into the affine input-to-hidden submapping A_{IH} and the nonlinear transformation $\bar{f}$

Mapping of a Three-Layer MLP Network

A mapping F is said to be a mapping of a three-layer *MLP* network defined as follows,

$$F = F_{HO} \circ F_{IH} \; (F : \Re^{N_I} \rightarrow \Re^{N_O}),$$

where N_I is the dimensionality of the input space and N_O is the dimensionality of the output space. F_{HO} is an affine mapping from N_H-dimensional subspace D^{N_H} of $\Re^{N_H}$ to $\Re^{N_O}$.

$$F_{HO} = (F_{HO_1}, \ldots, F_{HO_{N_O}}), F_{HO} : D^{N_H} \rightarrow \Re^{N_O}$$

$$F^{(p)}_{HO_k} = \sum_{j=1}^{N_H} w_{jk} F^{(p)}_{IH_j} - \Theta_{O_k}$$

where $F^{(p)}_{IH_k}$ is the output of the k-*th* hidden unit for the p-*th* training pattern, Θ_{O_k} is the threshold value ($\Theta_{O_k} \in R$) for the k-*th* output unit, w_{jk} is the real valued weight connection connecting the j-*th* hidden unit with the k-*th* output unit. F_{IH} is nonlinear multidimensional mapping

$$F_{IH} = \bar{f} \circ A_{IH} \qquad (F_{IH} : \Re^{N_I} \rightarrow D^{N_H}),$$

$$F^{(p)}_{IH_j} = f\left(\sum_{i=1}^{N_I} v_{ij} x^{(p)}_i - \Theta_{h_j}\right)$$

where Θ_{h_j} is the threshold value ($\Theta_{h_j} \in R$) for the j-*th* hidden unit, v_{ij} is the real valued weight connection connecting the i-*th* input unit with the j-*th* hidden unit, $x^{(p)}_i$ is the i-*th* coordinate of the p-*th* input vector $x^{(p)}$, $\bar{f}$ stands for a multidimensional nonlinear sigmoidal transformation in which each dimension of its N_H-dimensional domain vector is transformed by a sigmoidal transfer function f, ($\bar{f} : \Re^{N_H} \rightarrow D^{N_H}$), A_{IH} is an input-to-hidden affine submapping $A_{IH} : \Re^{N_I} \rightarrow \Re^{N_H}$.

Training in MLP Networks

Let T be a training set with cardinality N_P,

$$T = \{[x, y] \mid x \in \Re^{N_I} \wedge y \in \Re^{N_O}\}, \; |T| = N_P,$$

where each pair $[x, y]$ contains the input pattern x of the dimensionality N_I, and the expected output pattern y of the dimensionality N_O. Let u denote a set of free system parameters of a network (weights), $u = (w, h, v, \Theta)$, and the objective function E be defined as follows,

$$E(u, x) = \frac{1}{2N_P N_O} \sum_{p=1}^{N_P} \sum_{k=1}^{N_O} (F_k(u, x^{(p)}) - y^{(p)}_k)^2. \qquad (1)$$

Training in *MLP* networks is a process of minimization the objective function E,

$$\arg\min_u E(u, x),$$

given a finite number of samples $[x, y] \in T$ drawn from an arbitrary sample distribution.

Jacobean matrix, J_F, for a neural network is a matrix of the first derivatives of a network's mapping with respect to the free parameters. Analogously, error matrix, J_E, is a matrix of the first derivatives of an error function with respect to the free parameters. It can be expressed as a multiplication of the diagonal residual matrix Δ_O and Jacobean matrix J_F, $J_E = \Delta_O \cdot J_F$.

The task of rule extraction further requires to define classification and its realization by layered artificial neural networks.

Classification

Let $F = (F_1, \ldots, F_{N_O})$ be an arbitrary multidimensional mapping $F : \Re^{N_I} \rightarrow \Re^{N_O}$, and set $CLASS = \{CLS_1, \ldots, CLS_{N_O}\}$ be a set of classes $CLS_c \in \Re$. Classification CLS with respect to mapping F, $CLS(F)$, is defined as follows,

$$CLS : \Re^{N_I} \rightarrow CLASS$$

$$index(CLS_c) = index(F_c) \text{ such that}$$
$$F_c(x) = \max_l [F_l(x)], \; l = 1, \ldots, N_O,$$

where x is an input vector, F_l is the l-*th* coordinate mapping of F, and the function *index* returns an index of an indexed operand. The classification $CLS(F)$ defines a partitioning of the input space $\Re^{N_I}$ denoted as $PCLS(F)$.

Classification task, given by a training set T having N_O classes: $[CLS_1, \ldots, CLS_{N_O}]$, is performed by a neural network in such a way that each unit in the output layer is a representative of one class CLS_c, $c = 1, \ldots, N_O$. After presenting an input pattern, the strongest response in the output layer is selected as the classification answer of a network.

4. Dynamic Sample Selection

Dynamic sample selection techniques presented here focus on controlling the normed progressive search direction of the optimization technique by appropriate selection of exemplars. This enables determining an exemplar subset of training set that may vary at each iteration of optimization. Hence the name "dynamic sample selection" (DSS). Theoretical foundations of DSS have general validity. However, the specifics resulting from the use of first order line search optimization techniques and the type of the objective function are addressed.

4.1 General Functions & Dynamic Sample Selection

In this subsection the optimization case of general functions by first order line search techniques employing dynamic sample selection is addressed. The only requirement on optimized function E is the existence of the first partial derivatives. For this class of functions, practically very suitable expressions for dependencies of normed search directions on selected set of exemplars at each iteration of optimization procedure are shown. Theoretical details on derivation of the expressions can be found in [42] – [46].

Squared l_2 norm of the gradient is approximately expressed as,

$$\left\| \nabla E(u^{(k)}) \right\|_2^2 \approx \frac{(1-a)}{\left| \alpha^{(k)} \right|} \left| E(u^*) - E(u^{(k)}) \right|,$$

where a is a rate of convergence of steepest descent optimization technique, $\alpha^{(k)}$ is a scaling factor of the search direction at the state $u^{(k)}$, $\nabla E(u^{(k)})$ is a gradient vector at the given state $u^{(k)}$, and u^* is the optimum point.

The expressions for normed vector of search direction formulated above have particularly suitable form not only for the implementation purposes, but also for further analysis. The next intention is to observe dependence of squared l_2 norm of search direction on selected training set at a given step of learning:

$$\left\| \nabla E(u^{(k)}) \right\|_2^2 - \left\| \nabla E_{T^{(k)}}(u^{(k)}) \right\|_2^2 \approx \frac{(1-a)}{\left| \alpha^{(k)} \right|} \cdot \Xi$$

$$\Xi = [\, \left| E(u^*) - E(u^{(k)}) \right| - \left| E_{T^{(k)}}(u^*) - E_{T^{(k)}}(u^{(k)}) \right| \,], \quad (2)$$

where $\nabla E_{T^{(k)}}(u^{(k)})$ is a gradient vector at the state $(u^{(k)})$ for the selected set $T^{(k)}$, and $E_{T^{(k)}}(u^*)$ is a value of E at the optimum point u^* for the set $T^{(k)}$.

This underlines the fact that the diference of normed progressive directions of a neural network presented with the complete training set T and the selected training set $T^{(k)}$ at the step k is proportional to the residual expression (2). Proportionality is give by the scaling factor, $\frac{(1-a)}{\left| \alpha^{(k)} \right|}$. Implying from these theoretical results a general behavior of an arbitrary sample selection technique can be formulated:

For any convergent sequence $\{u^{(k)}\}$ of states of first order optimization technique such that $\{u^{(k)}\} \to u^*$, where u^* is the optimum point, holds: $\left\| s^{(k)}_{T^{(k)}} \right\|_2 \to \left\| s^{(k)} \right\|_2$.

The proof of the statement is detailed in [44]. Freedom of selecting exemplars dynamically at each iteration of training is controlled by the convergence of the optimization procedure. When the optimization procedure is relatively far from the optimum point the freedom for choosing proper exemplars in order to progress is higher. The amount of freedom decreases as the optimization procedure converges to the optimum point. That is, asymptotically l_2 norm of the search direction $s^{(k)}_{T^{(k)}}$ for selected data set $T^{(k)}$ should approach l_2 norm of the search direction $s^{(k)}$ for the complete data set T, as the algorithm reaches the optimum point. It is also important to note that the above statement holds for arbitrary dynamic sample selection mechanism, arbitrary first order optimization procedure, and arbitrary function E.

4.2 Functions having Lipschitz Continuous First Partial Derivatives & Dynamic Sample Selection

In the previous subsection the only assumption on functions to be optimized was the existence of first partial derivatives. The case addressed in this subsection imposes one more assumption on function E, that is, the Lipschitz continuity condition [47]:

For a given initial point $u^0 \in \Re^n$ if $E \in C^1$ on the set $S(u^0) = \{u \,|\, E(u) \le E(u^0)\}$, there exists a Lipschitz constant $K > 0$ such that,

$$\left\| \nabla E(u^{(i)}) - \nabla E(u^{(j)}) \right\|_2 \le K \cdot \left\| u^{(i)} - u^{(j)} \right\|_2$$

for every pair $u^{(i)}, u^{(j)} \in S(u^0)$.

Further focus is on deriving feasible expressions of normed gradient vectors allowing establishment of the dynamic sample selection approach for functions having Lipschitz continuous first partial derivatives. Similarly as in the previous case, the dependence of normed gradient vectors on the selected set of exemplars can be expressed using l_2 norms.

$$\left\| \nabla E(u^{(j)}) - \nabla E_{T^{(j)}}(u^{(j)}) \right\|_2 \ge$$
$$\ge K \cdot \frac{\left| E(u^{(i)}) - E(u^{(j)}) - E_{T^{(j)}}(u^{(i)}) + E_{T^{(j)}}(u^{(j)}) \right|}{\left\| \nabla E(u^{(k)}) \right\|_2^2}$$

Existence of stationary point u^* implies the following.

$$\left| 1 - \left\| \nabla E_{T^{(k)}}(u^{(k)}) \right\|_2 \right| \ge$$
$$\ge K \cdot \frac{\left| E(u^*) - E(u^{(k)}) - E_{T^{(k)}}(u^*) + E_{T^{(k)}}(u^{(k)}) \right|}{\left\| \nabla E(u^{(k)}) \right\|_2^2}$$

The above expressions are generally valid for any two points $u^{(j)}$, $u^{(j)}$, or the stationary point u^*. Their validity is also independent of optimization procedure. Hence they can be used for implementing dynamic sample selection mechanism into first order, second order, heuristic, or any other optimization procedure.

The relationship between the Lipschitz constant K and other parameters is formulated as,

$$\frac{1-a}{\left|\alpha^{(k)}\right|} \leq K.$$

This expression outlines important statement for practical implementation of dynamic sample selection mechanism into first order line search optimization procedures. It helps to simplify expressions for monitoring the dependence of the gradient vectors on selected set of exemplars.

4.3 Implementation and Simulations

A gradient vector ∇E in batch mode of back propagation procedure is formed as the sum of the gradients of the error function E for each exemplar presented to a network. Hence each presented exemplar plays a role in determining the search direction. By eliminating certain exemplars at a given step of training the search direction is modified. Proper modification of the search direction by exemplar selection may then be beneficial for the convergence speed increase of the optimization procedure.

Direct application of the derived theoretical material leads to the dynamic sample selection algorithm based on monitoring the values of the normed gradient vectors.

$$\min_{T^{(k)} \in T} \left[\left\| \nabla E(u^{(k)}) \right\|_2^2 - \left\| \nabla E_{T^{(k)}}(u^{(k)}) \right\|_2^2 \leq PI_1^{(k)} \right], \quad (3)$$

$$PI_1^{(k)} = \frac{(1-a)}{\left|\alpha^{(k)}\right|} \left| E(u^*) - E(u^{(k)}) \right|. \quad (4)$$

Similarly, dynamic sample selection approach can be derived for functions having Lipschitz continuous first partial derivatives.

$$\min_{T^{(k)} \in T} \left[\left\| \nabla E(u^{(k)}) - \nabla E_{T^{(k)}}(u^{(k)}) \right\|_2 \leq PI_2^{(k)} \right] \quad (5)$$

$$PI_2^{(k)} = \frac{K^{(k)}}{\left\| \nabla E(u^{(k)}) \right\|_2} \left| E(u^*) - E(u^{(k)}) \right|, \quad (6)$$

and

$$\min_{T^{(k)} \in T} \left[\left| 1 - \nabla E_{T^{(k)}}(u^{(k)}) \right\|_2 \right| \leq PI_3^{(k)} \Big], \quad (7)$$

$$PI_3^{(k)} = \frac{K^{(k)}}{\left\| \nabla E(u^{(k)}) \right\|_2^2} \cdot \left| E(u^*) - E(u^{(k)}) \right|, \quad (8)$$

where $K^{(k)}$ is a Lipschitz constant at the *k-th* iteration calculated as,

$$K^{(k)} = \frac{\left\| \nabla E(u^{(k)}) - \nabla E(u^{(k-1)}) \right\|_2}{\left\| u^{(k)} - u^{(k-1)} \right\|_2}$$

The expressions (3), (5), and (7) naturally follow from the presented theoretical material. However, they themselves require minimization process in order to find the proper set of selected exemplars $T^{(k)}$. The best solution to this problem would be to test all possible combinations on T. Apparently, this would result in computational excess of a method and impracticality for large data sets. To avoid the impracticality it is necessary to utilize a priori knowledge on importance of exemplars for progress of first order optimization techniques. Importance of a given exemplar is measured in terms of l_1 norm of $\nabla E^{(kp)}$, that is gradient of E for the *p-th* pattern at the k-th iteration.

Practically, in cases when optimization requires higher precision, asymptotic behavior is not always satisfied due to the approximations introduced in terms $PI_1^{(k)}$ (4), $PI_2^{(k)}$ (6), and $PI_3^{(k)}$ (8). Then dynamic sample selection may have divergent behavior and may eliminate large amount samples even if the algorithm has reached the attractor basin.

For this reason, the suppression function is introduced.

$$N_{EP} = (N_P - N_{TP}) \frac{\frac{1}{Iter} \sum_{j=1}^{Iter} PI_i^{(j)}}{PI_i^{(1)}} \quad (9)$$

N_{EP} is the maximum number of eliminated exemplars, N_P is the cardinality of the data set T, N_{TP} is the pre-determined minimum number of exemplars in order to keep the optimization problem well-posed, *Iter* is a given iteration, $PI_i^{(j)}$, i = 1, 2, 3, is one of functions (4), (6), or (8).

The efectiveness of the dynamic sample selection algorithm implemented into the back propagation training algorithm is demonstrated on simulations. Simulation tasks are selected according to the value of E-FP (exemplars & free parameters) ratio. First, performance of dynamic sample selection algorithm is tested on the problem with E-FP ratio 1.66. The second simulation task has high value of E-FP ratio equal to 15.

In the case of E-FP ratio 1.66 the network had configuration 4-3-1. The network was trained on the Lenses data set [48]. The hidden layer of the network contained sigmoidal transfer function units. The network was trained on the Lenses data set. The stopping criterion for network's training was the value of the expected error less than or equal to 0.05. The weights of the network were initialized randomly in the interval $<-0.1, 0.1>$ which corresponded to the steepest part of the sigmoidal nonlinearity in the hidden units. It was observed that the implementation of dynamic sample selection algorithm, based on (3) and (4) into the back propagation training procedure results in 32.06 % overall increase of convergence speed measured in number of training cycles required to reach the given value of the expected error.

In the second case we observed the simulation results for the problem that had E-FP ratio equal to 15, which is extremely high. The network had configuration 4-2-1 with sigmoidal hidden units. The training set was the IRIS data set [49], [50]. Network's free parameters were again initialized as random values in the interval $<-0.1, 0.1>$. Training was terminated when the value of the expected error decreased below 0.013. Overall increase of convergence speed of 4.03 % is indicated for implementation of dynamic sample selection based on (3), (4), and 3.27 % for dynamic sample selection implementation utilizing expressions (5), (6).

5. Parameter Adaptation

This section introduces modification of the first order line search optimization technique with automatically and dynamically adaptable parameters. The focus is on dynamic adjustments of both step length $\alpha^{(k)}$ and momentum term $\beta^{(k)}$. Step length $\alpha^{(k)}$ and momentum term $\beta^{(k)}$ are allowed to take diferent values at each iteration of the optimization procedure [51] - [56].

First observed is the extension of first order line search optimization procedure with adaptable step length $\alpha^{(k)}$ to a procedure that additionally incorporates constant momentum term β. Initially, the following expressions are obtained [56]:

$$|\alpha^{(k)}| \geq \left((1-a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + \right.$$

$$\left. + |\beta| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right), \tag{10}$$

and

$$|\alpha^{(k)}| \leq \left((1+a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + \right.$$

$$\left. + |\beta| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right), \tag{11}$$

where $s^{(k-1)}$ is a search direction at the $(k-1)$-*th* iteration of the optimization procedure.

Assumption of superlinear convergence rates results in the following update formula for step length $\alpha^{(k)}$.

$$|\alpha^{(k)}| = \left(\frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + |\beta| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right) \tag{12}$$

Superlinear convergence rates assumption shrinks the boundaries (10) and (11) for step length $\alpha^{(k)}$ to a single point. This naturally simplifies the line search subproblem to a one step calculation of $\alpha^{(k)}$. In order to attain higher flexibility, it is also recommended to use the median value of (10) and (11) for calculation of the step length $\alpha^{(k)}$,

$$|\alpha^{(k)}| = \left(a \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + |\beta| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right) \tag{13}$$

where a is a parameter determined by user. The dependency of the modifiable momentum term $\beta^{(k)}$ can be obtained from first order line search optimization techniques, and conjugate gradient techniques in particular. Considering the definition of general linear convergence rates and essential formula for parameter updates of conjugate gradient methods [57], the following inequalities are derived [56].

$$|\beta^{(k)}| \geq \frac{1}{\|s^{(k-1)}\|_2} \left(|\alpha^{(k)}| \cdot \|\nabla E(u^{(k)})\|_2 - \right.$$

$$\left. - (1+a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2} \right) \tag{14}$$

$$|\beta^{(k)}| \leq \frac{1}{\|s^{(k-1)}\|_2} \left(|\alpha^{(k)}| \cdot \|\nabla E(u^{(k)})\|_2 - \right.$$

$$\left. - (1-a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2} \right) \tag{15}$$

From the assumption of superlinear convergence rates of conjugate gradient method the following implies.

$$|\beta^{(k)}| = \frac{1}{\|s^{(k-1)}\|_2} \left(|\alpha^{(k)}| \cdot \|\nabla E(u^{(k)})\|_2 - \right.$$

$$\left. - \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2} \right) \tag{16}$$

To determine the dependencies of step length $\alpha^{(k)}$ in the conjugate gradient method it is possible to apply the formerly obtained results. Then for the step length $\alpha^{(k)}$, considering the dynamically adjustable momentum term $\beta^{(k)}$,

$$|\alpha^{(k)}| \geq \left((1-a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + \right.$$

$$\left. + |\beta^{(k)}| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right), \tag{17}$$

and

$$|\alpha^{(k)}| \leq \left((1+a) \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + \right.$$

$$\left. + |\beta^{(k)}| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right), \tag{18}$$

is implied. Accounting for the superlinear convergence rates of the conjugate gradient method,

$$|\alpha^{(k)}| = \left(\frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2} + |\beta^{(k)}| \cdot \frac{\|s^{(k-1)}\|_2}{\|\nabla E(u^{(k)})\|_2} \right) \tag{19}$$

is obtained.

As clearly seen from (14)-(16) and (17)-(19) the expressions for the adjustable momentum term $\beta^{(k)}$ (14)-(16) incorporate step length $\alpha^{(k)}$, likewise the expressions for adjustable step length $\alpha^{(k)}$ (17)-(19) contain momentum term $\beta^{(k)}$. This in practice leads to the dynamic loop. Thus it is impossible to determine which expressions should be calculated first (whether the ones for $\alpha^{(k)}$, or the ones for $\beta^{(k)}$). To overcome this difficulty it is necessary to find another relevant expression for either the adjustable momentum term $\beta^{(k)}$ or the adjustable step length $\alpha^{(k)}$. Theoretical material in [51] already offers possible solution to this problem in the form of the expressions,

$$|\alpha^{(k)}| = a \cdot \frac{|E(u^*) - E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2}, \tag{20}$$

$$|\alpha^{(k)}| = a \cdot \frac{|E(u^{(k)})|}{\|\nabla E(u^{(k)})\|_2^2}. \tag{21}$$

However, it can easily be verified that the substitution of (20) into (16) leads to $\beta^{(k)} = 0$, which on one side supports the

sufficiency of only adjustable step length $\alpha^{(k)}$, but on the other side, it eliminates the momentum term. Hence the only appropriate choice for $\alpha^{(k)}$ is expression (21). Then, substitution of (21) into (16) results in the expression for the adaptable momentum term $\beta^{(k)}$ as follows,

$$\left|\beta^{(k)}\right| = \frac{a}{\left\| s^{(k-1)} \right\|_2 \cdot \left\| \nabla E(u^{(k)}) \right\|_2} \cdot$$

$$\cdot \left(E(u^{(k)}) - \left| E(u^*) - E(u^{(k)}) \right| \right) \qquad (22)$$

Taking into account the absolute value $\left| E(u^*) - E(u^{(k)}) \right|$ in (22) the following two expressions for the modifiable momentum term further imply,

$$\left|\beta^{(k)}\right| = a \cdot \frac{E(u^*)}{\left\| s^{(k-1)} \right\|_2 \cdot \left\| \nabla E(u^{(k)}) \right\|_2}, \qquad (23)$$

and

$$\left|\beta^{(k)}\right| = a \cdot \frac{2E(u^{(k)}) - E(u^*)}{\left\| s^{(k-1)} \right\|_2 \cdot \left\| \nabla E(u^{(k)}) \right\|_2}. \qquad (24)$$

It is important to note that the convergence proof in [56] further eliminates expression (24). Use of expression (24) results in the divergence of the optimization technique. Then the only suitable expression for adaptable momentum term $\beta^{(k)}$ is (23). The constant a stands for the universality of the algorithm. This leads to the following modification of the conjugate gradient optimization technique.

ALGORITHM 1

1. Set the initial parameters: $u^{(0)}$, $E(u^*)$, (β).
2. Calculate the gradient $\nabla E(u^{(k)})$.
3. *Constant momentum:* calculate $\alpha^{(k)}$ according to expression (12) or (13).
   Adaptable momentum: calculate $\alpha^{(k)}$ according to (21) and $\beta^{(k)}$ as (23).
4. Update the system parameters as follows.

$$u^{(k+1)} = u^{(k)} - \alpha^{(k)} \cdot \nabla E(u^{(k)}) + \beta^{(k)} \cdot s^{(k-1)}.$$

ALGORITHM 1 has substantially simplified the line search subproblem (step 3). The proper parameters $\alpha^{(k)}$ and $\beta^{(k)}$, (β), are determined automatically in a single calculation. ALGORITHM 1 has a linear computational complexity $O(N_F)$, where N_F is a number of free parameters. The necessity of keeping the track of the previous search direction $s^{(k-1)}$ in conjugate gradient techniques leads to the linear memory requirements $O(N_F)$ of ALGORITHM 1. Despite the simplicity of the line search subproblem, ALGORITHM 1 is convergent with superlinear convergence rates [56]. The superlinear convergence rates are established under the assumption that second and higher order terms of Taylor expansion of the objective function E around the optimum point are negligible for convergence of the sequence of points $[u^{(k)}]_k$ generated by ALGORITHM 1.

ALGORITHM 1 is demonstrated in Fig. 3. ALGORITHM 1 (charts *b)* and *c)*) clearly converges substantially more smoothly

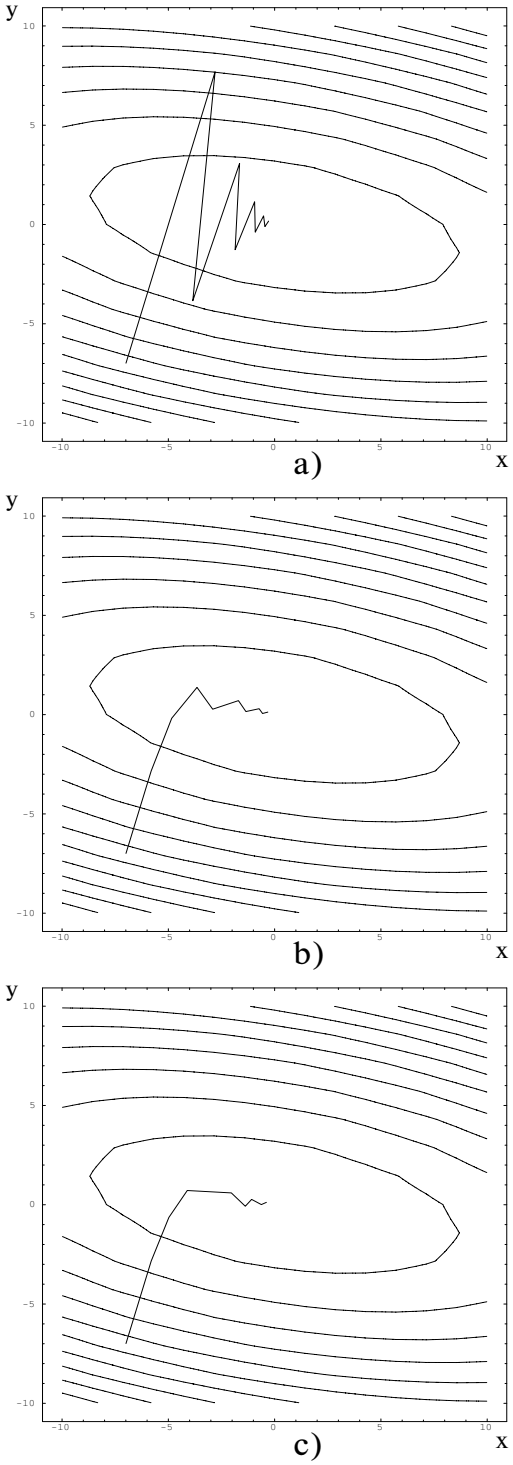

Fig. 3 Comparison of optimization progress between ALGORITHM 1 (chart b) (constant β = 0.1) and c) (adjustable α, β)) and BP with momentum (α = 0.3, β = 0.1) (chart a)) on quadratic function f(x, y) = $= 0.5x^2 + 3y^2 + xy$ from the starting point [-7,-7]. ALGORITHM 1 had setting: E(u) = 0, a = 1. Stopping criterion was the value f(x, y) ≤ 0.1. It is evident that the progress of ALGORITHM 1 is smoother and faster than BP with momentum.*

to the optimum point than the conventional conjugate gradient method (chart *a)*) having a constant step length and momentum.

5.1 Simulations

In this subsection the effectiveness of the algorithms is practically demonstrated on five tasks represented by the following data sets: Lenses [48], Glass, Monks 1 [58], Monks 2 [58], and Monks 3 [58]. The presented algorithms were applied to training various MLP networks to perform tasks given by five, the above mentioned, data sets. Neural network's performance was optimized according to the mean square error. The stopping criterion was the value of the expected error.

In the case of the Lenses data set [48], a neural network had configuration 4-3-1 with sigmoidal hidden units. Expected error was set to $5 \cdot 10^2$. In the Glass problem, a network was configured as: 9-5-1 (sigmoidal hidden units) and the expected error was equal to 0.35. Finally, for Monks 1, 2, and 3 problems [58] a neural network structure was set as: 6-3-1 (sigmoidal hidden units), and the expected error was equal to 0.103. Network's weights were initialized randomly in the interval $< -0.1, 0.1 >$, which corresponded to the steepest region of the sigmoidal transfer function of the hidden units. The parameter a was equal to 1. In case network's error did not converge to the value less than or equal to the expected error within 20000 cycles, the training process was terminated. It is interesting to note that additional stopping condition of maximum 20000 cycles was practically applied only to the BP employing standard first order techniques. ALGORITHM 1 always converged.

The experiments were performed with the value of the step length (learning rate) for BP corresponding to the best results of BP as reported in [51] (in Monks 1 case $\alpha = 0.8$, and for all other data sets $\alpha = 0.9$). The momentum term ranging from 0.1 to 0.7 in 0.1 increments was then applied. BP with the momentum term and the best value of step length (denoted in further text as BPM) was compared to ALGORITHM 1 with constant momentum term (see results in Table 1), and with automatically adjustable momentum term (see Table 2). Values of the constant momentum term in ALGORITHM 1 were set equal to the values of the momentum term in BPM. For a given setting of learning rate and momentum term the simulations were run 10 times for different randomly initialized weights in the interval $< -0.1, 0.1 >$. Percentual convergence speed increases were calculated in order to simplify the comparison. Hence the values in Table 1, and 2 represent ten-run-averages. Criterion for comparison of the convergence speed was the number of cycles required to decrease the mean square error E of a neural network below the value of the expected error.

It is clear, from Table 1, and 2, that the proposed algorithm (that is, ALGORITHM 1) converged substantially faster than the standard techniques. As previously mentioned, ALGORITHM 1 converged each time, whereas BPM for some initial setting of weights and parameters α, β could not achieve convergence even after 20000 cycles. This accounts for higher stability of ALGORITHM 1.

Table 1: Comparison of ALGORITHM 1 (constant momentum) and BPM with setting of constant learning rate that corresponded to the best obtained results of BP. Momentum term was set from 0.1 to 0.9 in 0.1 increments, and kept constant for both ALGORITHM 1 and BPM. ALGORITHM 1 had the following setting: $E(u^*) = 0$, $a = 1$. Values in the table represent ten-run-averages of percentual convergence speed increase of ALGORITHM 1 over BPM. Convergence speed was compared in terms of cycles required to reach a given value of the expected error.

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	AVR
Lenses (α = 0.9)	43.99	19.65	7.03	44.49	93.12	94.46	95.14	56.84
Glass (α = 0.9)	54.4	52.05	47.76	44.43	39.02	35.43	29.05	43.18
Monks 1 (α = 0.8)	57.37	51.73	40.42	39.7	49.38	85.07	90.95	59.23
Monks 2 (α = 0.9)	-12.41	-28.49	4.55	58.95	86.87	95.18	96.07	42.96
Monks 3 (α = 0.9)	40.91	34.33	31.97	58	82.07	87.39	89.7	60.62

Table 2: Comparison of ALGORITHM 1 (adaptable momentum) and BPM with learning rate setting corresponding to the best obtained results of BP. Momentum term setting varied from 0.1 to 0.9 in 0.1 increments. ALGORITHM 1 used the setting: $a = 1$. Values in the table stand for ten-run-averages of percentual convergence speed increase of ALGORITHM 1 over BPM. Convergence speed comparison was made in terms of number of cycles necessary to decrease the network's error below the value of the expected error.

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	AVR
Lenses (α = 0.9)	51.23	20.5	10.14	46.95	93.18	95.15	97.13	59.18
Glass (α = 0.9)	55.35	53.38	48.93	45.97	41.35	37.89	30.25	44.73
Monks 1 (α = 0.8)	59.37	53.25	43.19	41.53	53.68	87.79	93.25	61.72
Monks 2 (α = 0.9)	1.45	-3.47	14.33	60.78	87.96	97.31	97.93	50.89
Monks 3 (α = 0.9)	42.35	40.37	35.96	59.97	83.47	89.93	92.23	63.47

6. Structural Adaptation

The progress of a training algorithm can be seen as a movement in a space with principal coordinates defined by the singular vectors. The error surface increases most rapidly in the direction of a vector corresponding to the maximum singular value and most slowly in a direction of a vector corresponding to the minimum singular value. The singular values give partial information about the determination of the next step of an algorithm. They provide a relative measure on the direction and the proportion of the movement. Since the singular value decomposition is computationally expensive task it is desirable to approximate location of the singular values given by the spectral radius estimate [31].

Spectral Radius Estimate
Let $A \in M_{m,n}$. For the spectral radius of singular values holds,

$$\rho(A) \leq min[\rho'(A), \rho''(A)],$$

where

$$\rho''(A) = min\left\{ \max_i \left(\sum_{j=1}^{m} |a_{ij}| \right), \max_j \left(\sum_{j=1}^{n} |a_{ij}| \right) \right\},$$

$$\rho'(A) = min\left\{ \max_r \left(\sqrt{\sum_{c=1}^{m} \left| \sum_{j=1}^{n} a_{rj} \cdot a_{jc} \right|} \right), \max_c \left(\sqrt{\sum_{r=1}^{m} \left| \sum_{j=1}^{n} a_{rj} \cdot a_{jc} \right|} \right) \right\}.$$

"r" and "c" denote row and column indices of AA^T, respectively.

How the estimates ρ'' (25) and ρ' (25) typically work in practice is shown in Fig. 4. The task depicted in Fig. 4 is training an MLP neural network with the configuration 2-2-1 on the XOR problem. Network's connections were initialized by the exponential series with base 0.9 and 0.5 for hidden-to-output and input-to-hidden weights, respectively. Steepest descent version of BP with the constant learning rate 0.9 was used. Training was terminated when the mean square error decreased below 10^{-2}. The actual spectral radius ρ in Figure 4 was obtained by the singular value

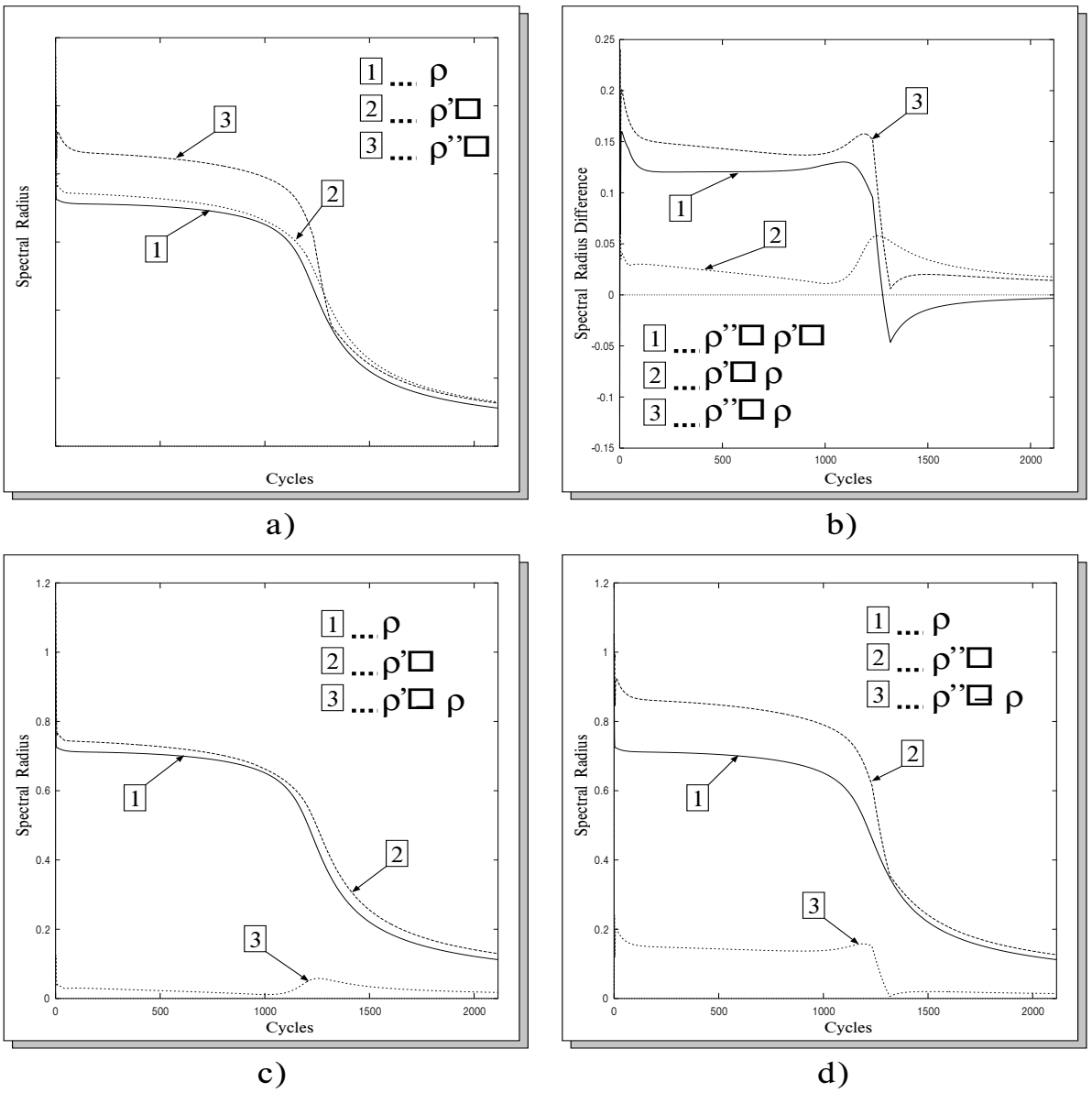

Fig. 4 Typical behavior of the estimates of spectral radiuses ρ' and ρ''. The spectral radius ρ was obtained by singular value decomposition. A training task for MLP neural network with the configuration 2-2-1 was the standard benchmark XOR. Chart a) shows the values of ρ, ρ', and ρ'' at each cycle of training. Chart b) displays the differences $\rho'' - \rho'$, $\rho' - \rho$, and $\rho'' - \rho$. Details of the relationship between ρ and ρ' is shown in chart c). Analogously, the values of the estimate ρ'' in relation to ρ are depicted in chart d).

decomposition. Note that the XOR problem serves as a testbed for essential linear non-separability. The training set contains four patterns $T = \{[(0, 0), 0]; [(0, 1), 1]; [(1, 0), 1]; [(1, 1), 1]\}$. A sufficient three layer network configuration incorporates two input units, two hidden units, and one output unit.

The reason for having two different expressions for spectral radiuses of singular values and taking minima of them follows from the need for more precise location and from the nature of the optimization techniques. When the first order optimization procedure is relatively far from the equilibrium point, it is more likely that ρ'' gives more precise estimate of the spectral radius because the derivatives will be higher in value. However, when the algorithm converges (in an ideal case to 0) then also the first order derivatives should converge, possibly to 0, and thus become smaller. In this case, when the derivatives are lesser than 1, the expression ρ' may locate the singular values within a smaller interval.

As it can be seen from the charts a) and b), the estimate ρ' was more precise when the network was relatively far from the optimum point. Once the attractor basin was reached, the estimate ρ'' showed slightly higher precision than ρ'. The reason for this is the difference of the slopes of the error surface. After the initial progress (approximately 40 cycles) the network was progressing on the flat region of the error surface for almost 1200 cycles. Surface flatness is indicated by small gradient values. Therefore, the estimate ρ' had smaller values than ρ''. As the algorithm reached the attractor basin, the network started to progress on a sharper slope of the error surface. The gradients were higher. Thus the estimate ρ'' had smaller values than ρ'. When the algorithm converged to the optimum point, both ρ' and ρ'' converged to almost the same values (see chart a) and also differences in chart b)). Specifics of each estimate ρ' and ρ'' in relation to ρ are depicted in charts c) and d).

Taking into account the static and dynamic importance of the weight connections in the network the authors propose the measure of use of the network's potentials (to progress) at each step of a training algorithm as,

$$PM = \frac{1}{\rho(J_E)} \frac{1}{N_F} \sum_{u_l \in u} \left| u_l \sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right|, \qquad (25)$$

where E stands for some specific error function, u_l is the *l-th* free parameter of a network, and $\rho(J_E)$ is the estimate of a spectral radius of the error matrix J_E. The expression (25) represents the overall average performance of a network.

The value of the estimate of a spectral radius cannot exceed the minimum of the maxima of row and column sums of the elements in an error matrix for an artificial neural network. That is, the largest singular value σ_{max} lies within the interval specified by the matrix measures (or norms) such as column or row l_1 norms. Regarding search directions, a network's progress in each dimension is limited by the value of the estimate of a spectral radius. The sum of column elements of a specific matrix (depending on error function) represents how far a network will move in a particular dimension. Hence, the network in any dimension cannot progress more than the estimate shows. Considering the static relevance of weight $u_l \in u$, given by its real value ($u_l \in \Re$), multiplied by the dynamic relevance of the connection represented by the sum of the gradients for each pattern sample $\left(\sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right)$ and then scaling the multiplication with respect to the maximum progress formulated by the estimate of a spectral radius ρ, the overall average performance of a network is obtained at each iteration (25). From (25) immediately implies that the individual performance of each connection is given as,

$$I_{pm_{u_l}} = \frac{1}{\rho(J_E)} \left| u_l \sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right|. \qquad (26)$$

Similarly the maximum performance of a network during training can be measured,

$$PM_m = \frac{1}{\rho(J_E)} \max_{u_l \in u} \left| u_l \sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right|. \qquad (27)$$

The relevance and importance of the above measures is illustratively demonstrated on the standard benchmark task XOR.

First, the relevance of measures (25) and (27) is demonstrated. To obtain a closer look at the behavior of a training procedure the auxiliary expressions are evaluated,

$$D_{pm} = \sum_{u_l \in u} \left| u_l \sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right|, \qquad (28)$$

$$D_{pm_m} = \max_{u_l \in u} \left| u_l \sum_{p=1}^{N_p} \frac{\partial E^{(p)}}{\partial u_l} \right|. \qquad (29)$$

Configuration of the network was 2-2-1 and the weight connections were initialized by the exponential series with base 0.9 for the hidden-to-output weights and 0.5 for the input-to hidden weights. A batch mode of BP training with a constant learning rate equal to 0.9 was used. Simulation charts are shown in Figure 5 and Fig. 6. Fig. 5 shows the performance of a network according to the performance measure (25) and Figure 6 depicts the maximum performance chart for a network according to the measure (27). Both figures clearly indicate four distinguishable phases of a training procedure.

The first phase depicts the initial progress of a network and sudden stagnation. The error E decreased in a few starting training cycles and then stabilized. This phase is indicated by a rapid decline of both the average performance PM (25) and the maximum performance PM_m (27), which exactly reflects the fact that the network was not progressing, since it has reached the flat surface of an error landscape. The estimate of a spectral radius $\rho(J_E)$ had lightly fluctuated around 1 and then stabilized. Detail of this phase is given in upper charts of Fig. 5 and Fig. 6. It shows first 40 cycles of a training procedure. Low levels of D_{pm} and D_{pm_m} measures underline the fact that the network reached a flat region of error surface which is, according to the error E, positioned quite high from the minima.

The second phase shows slow progress of a network on a flat region of an error surface. The error E changes very little. The

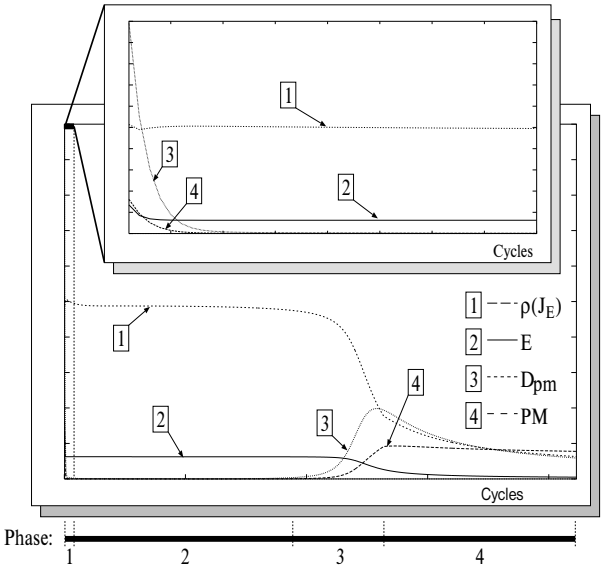

Fig. 5 Training behavior of the network with configuration 2-2-1 for the XOR problem according to the measures PM and D_{pm}. Four training phases could be distinguished. The detail of the first phase is given in the upper chart. The network converged to the point where $E < 10^{-2}$ after 2114 cycles. A batch mode of BP training used constant learning rate equal 0.9.

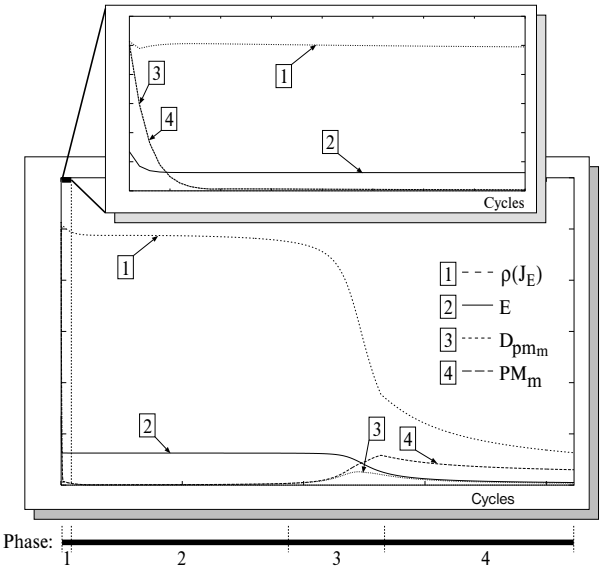

Fig. 6 Behavior of the network with configuration 2-2-1 according to the measures PM_m and Dp_{m_m} during training to perform the XOR task. Similarly as in the previous figure there are four distinguishable phases. The first phase is shown in detail in the upper chart. The error $E < 10^{-2}$ was reached after 2114 cycles for a batch mode of BP training with constant learning rate 0.9.

performance of a network (by means of the measures *PM* and PM_m) is very low. The estimate of a spectral radius $\rho(J_E)$ based on l_1 norms is relatively stable and high. High values of the error E and the estimate of a spectral radius $\rho(J_E)$, and low values of the expressions D_{pm} and D_{pm_m} imply that the flatness of an error surface is caused by sign oscillations of the weight gradients for different patterns.

In the third phase, the network passed over the at region of an error surface and reached the basin of attraction of an attractor point. It started to converge toward the minimum. The error *E* smoothly decreased and according to the performance measures *PM* and PM_m the network indicates rapidly increased progress. Declining estimate of a spectral radius $\rho(J_E)$ and rising values of D_{pm} and D_{pm_m} indicate that the network balanced its computational resources by eliminating the oscillations of gradients for different training samples. A sharp rise of D_{pm} and D_{pm_m} indicates that the network progressed on a sharp slope of an error surface.

The fourth phase shows the smooth convergence of a network toward the minimum point. Error nicely decreases and the network progresses at constant rate, as indicated by the average performance measure *PM*. The maximum performance PM_m smoothly stabilizes. The stable convergence ratio is also nicely indicated by stabilization of the estimate of a spectral radius as well as measures D_{pm} and D_{pm_m}. Low values of maximum gradients (D_{pm_m}) underline the reach of the optimum point.

6.1 Structural Adaptation Utilizing Learning Performance Measures

The relevance and importance of the derived performance measures for structural modifications of a network, particularly pruning, is shown. As previously mentioned, the expression (26) represents a combination of static and dynamic importance of a specific weight connection in a structure of a network. Thus it can be used for detecting less important structural elements suitable for pruning. The use of the measure (26) for pruning is illustratively depicted again on an example of the XOR problem (see Fig. 7).

An overdetermined network structure with configuration 2-3-1 was generated. The initial values of the weights were set in the same way as in the previously mentioned simulation. A batch mode of BP training procedure with the constant learning rate 0.9 was used. It can be seen that the original network structure (2-3-1) was able to converge approximately after 1900 cycles (to be precise 1896 cycles) to the point where the error $E < 10^{-2}$. The evidence that the network converged is shown in upper-left chart of Fig. 7. After approximately 1000 cycles the network started to converge. The performance measure *PM* as well as D_{pm} rose. The estimate of a spectral radius started to decline. And from around cycle 1400 the network was progressing at a constant rate, as indicated by the stabilized curve of the *PM* measure.

When the network reached cycle 111 the maximum performance measure PM_m indicated minimum value ($1.229526 \cdot 10^{-3}$). At this point, since the progress was very slow, the network was

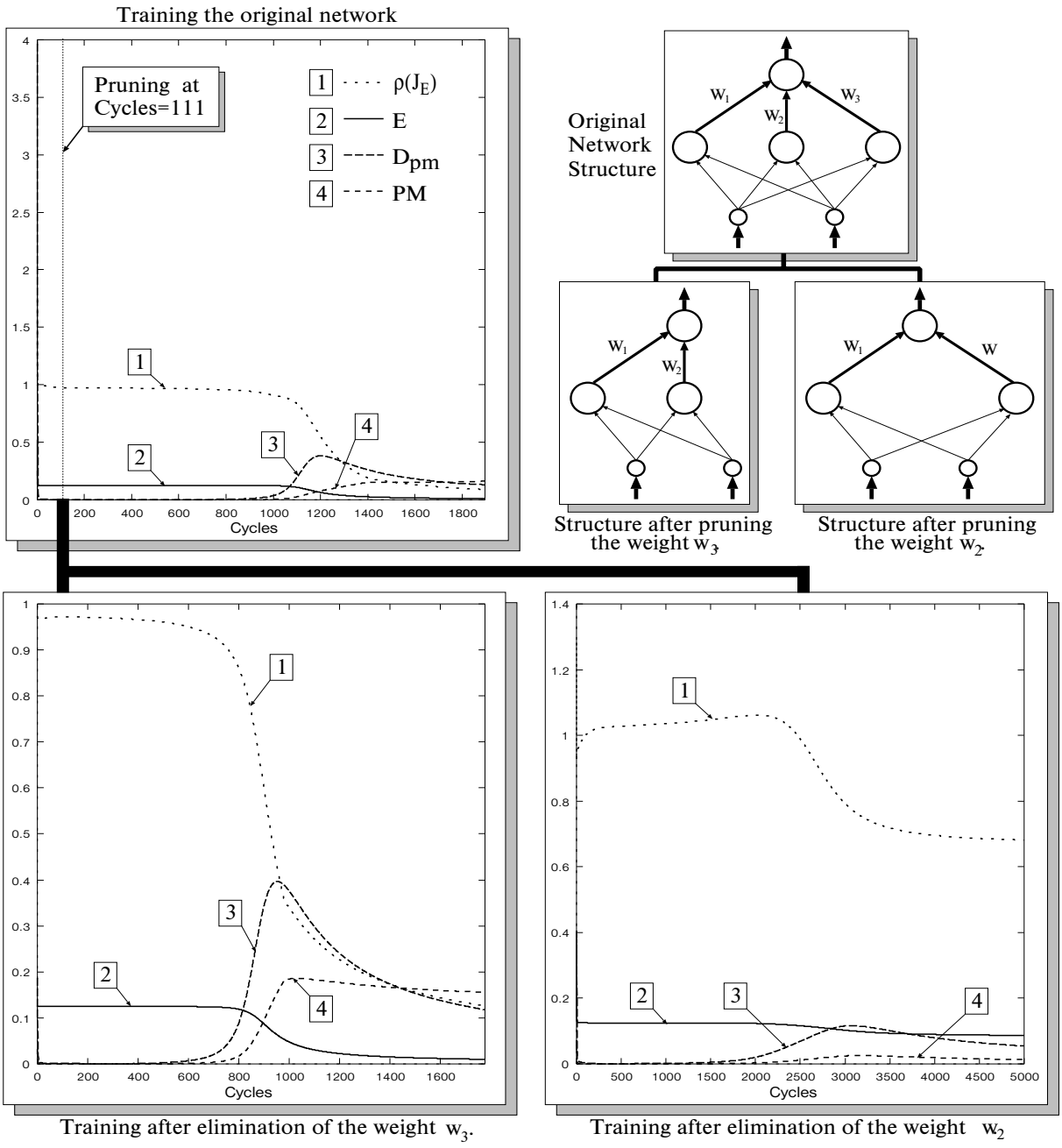

Fig. 7 The effect of pruning the original network with configuration 2-3-1 on training behavior. The pruning criterion was $I_{pm_{u_j}}$. Top-left chart depicts the training behavior of the original network (2-3-1). Top-right part shows the modification of the structure after elimination of the hidden-to-output weight w_3 (left) and after elimination of the hidden-to-output weight w_2 (right). Bottom-left chart shows the training behavior of the network after pruning the hidden-to-output weight w_3 (the weight with low individual performance measure). Bottom-right chart shows the training behavior of the network after pruning the hidden-to-output weight connection w_2 (the weight with high individual performance measure). The pruning point for the original network structure was cycle 111.

suitable for pruning. The individual performance measures (26) of hidden-to-output weights were evaluated. It was found that at the cycle=111 the individual performance measure for the hidden-to-output weight w_3 was extremely low ($1.5993 \cdot 10^{-5}$), which was about ten times less than that for the hidden-to-output weight w_2 ($1.20916 \cdot 10^{-4}$). The individual performance measures for the hidden-to-output weights w_2 and w_3 were at the same level (approximately ten times higher than $I_{pm_{w_3}}$). Hence the suitable candidate for pruning was the weight w_3. The bottom-left chart in Figure 7 shows the training behavior of the network after pruning

the hidden-to-output weight w_3. Six hundred cycles after pruning the weight w_3 the network started to converge (see the charts of measures *PM*, $\rho(J_E)$, and D_{pm}). And 1000 cycles after pruning the network was progressing at almost constant rate (see the curve *PM*). At cycle 1777 after pruning the network reached a state where the error E was less than 10^{-2}.

It is also important, on the other hand, to observe the behavior of a network when the weight with a high individual performance measure (26) was pruned. The bottom-right chart of Figure 7 shows the training behavior of the network after pruning the hidden-to-output weight connection w_2, that is, the connection with high individual performance measure (26). It is clear (from the bottom-right chart in Figure 7) that the network after pruning the hidden-to-output weight w_2 was not able to converge even after 5000 cycles. Approximately 2000 cycles after pruning the performance *PM* of the network slightly rose, however, after reaching the maximum point at cycle=2995 ($2.036367 \cdot 10^{-2}$) the performance *PM* declined. The estimate of a spectral radius $\rho(J_E)$ dropped, but again stabilized at a relatively high level (around 0.7). The measure D_{pm} rose, but then decreased. From the curves of $\rho(J_E)$, *PM*, and D_{pm} it can be concluded that the network reached another relatively flat region of the error surface. Generally, the performance of the network was very low during the whole period of 5000 cycles after pruning the hidden-to-output weight w_2. This underlines the relevance of the individual performance measure (26) as well as the other measures (25) and (27).

It is clear from the above and from (25), (26), and (27) that the essential reference ground for comparison of the combined static and dynamic importance of the network elements is the estimate of a spectral radius $\rho(J_E)$. It is the only suitable reference ground directly implied from the nature of the first order optimization procedures. Furthermore, it has three remarkable properties:

It is dynamic. The estimate of a spectral radius $\rho(J_E)$ dynamically changes according to the state of a training procedure. Thus it represents the most suitable dynamic reference ground at every step of training. This naturally overcomes the problems of choosing the right reference value, which is essential for regularization based approaches (e.g. choosing the right preassigned parameter u_0 in the weight elimination procedure). Most likely there is a very small class of problems which satisfies the optimum condition for constant reference ground.

It is sensitive. The estimate of a spectral radius $\rho(J_E)$ is sensitive enough to reflect the occurrences leading to the progress of a network.

It can be manipulated. Structural modifications and sample selection can influence the values of the estimate of a spectral radius $\rho(J_E)$ so as to maximize the expressions (25), (26), and (27).

The third point, the ability to manipulate the estimate of a spectral radius $\rho(J_E)$, is of primary interest in this study. Although there have been a variety of modifications of first order optimization techniques and their implementation in the field of artificial neural networks, a deterministic theoretical analysis is still lacking. This is true particularly for cases where the training algorithms incorporate dynamic structural changes. Hence, it is the researcher's intention to shed some light onto this specific area of a neural network field.

7. Rule Extraction

The rule extraction problem from neural networks is addressed in its principal form. That is, given an arbitrary three layer artificial neural network trained on particular data extract the rules that reflect the network's data classification as correctly as possible. Note that this is general and essential principle of rule acquisition. The only choice of three layer neural networks is due to their universal approximation abilities. Methodology introduced here is independent of rule types, network types, and learning strategy. Hence, there is no assumption of any a priori knowledge implementation into a neural network.

Given a mapping F of three layer artificial neural network trained on the classification task described by a training set T, the primary interest is to derive the rules that classify the data as correctly as neural network. This is the underlining problem of rule extraction from trained neural networks. Sometimes there must not be the solution to this problem, however, the classification rules describing data with certain accuracy can still be obtained. The availability of solution to rule extraction problem is stated in the following.

Crisp Rule Extraction

Let bu be a set $bu = \{1, 0, 1\}$ (or $bu = \{1, 1\}$). Let $F = F_{HO} \circ F_{IH}$ be a mapping of a nonlinear neural network with structure $N_I - N_H - N_O$ such that input-to-hidden weights v_{ij} and hidden-to-output weights w_{jk} are real valued parameters, $v_{ij} \wedge w_{jk} \in \Re$, and $N_H \geq [\log_{|bu - \{0\}|}(N_O)]$. If there exists a set of vectors $[cr^{(1)}, \ldots, cr^{(N_P)}]$ such that,

$$\max_{k=1,\ldots,N_O} [w_k^T \cdot F_{IH}^{(p)}] = \max_{k=1,\ldots,N_O} [\mathrm{b}w_k^T \cdot (F_{IH}'^{(p)} + cr^{(p)})],$$

$$F_{IH}'^{(p)} = f(bv, x^{(p)});\ cr^{(p)} \in [cr^{(1)}, \ldots, cr^{(N_P)}],$$

where $bw_k = (bw_{1k}, \ldots, bw_{N_H k})$, $bw_{jk} \in bu$, and $bv = (bv_{11}, \ldots, bv_{N_I N_H})$, $bv_{ij} \in bu$, there exists the representation of the network's mapping in form of the rules,

$$IF \left(\bigwedge_{i=1,\ldots,N_I} LOx_i^{(k)} \mathcal{E} < x_{iS}^{(k)}, x_{iE}^{(k)} > \right) THEN\ CLS_k, \qquad (30)$$

where $k = 1, \ldots, N_O$, LO is either void or logical operator of negation $\neg$, and $\mathcal{E}$ is either $\in$ or $\notin$. The rules (30) classify the patterns $[x^{(p)}, y^{(p)}] \in T$, $p = 1, \ldots, N_P$, as correctly as neural network.

General proof of the above statement is provided in [38]. Practical implications of the constructive proof lead to the methodology for deriving the rules from arbitrary three layer neural network. The methodology is applicable even in the cases

when the theoretical conditions are not strictly satisffed. It can be described in the following 5 steps.

1) Code the weight vectors w_k, v_j into trinary vectors bw_k, bv_j according to the shortest Euclidean distance ($min\ d(w_k, bw_k)$, and $min\ d(v_j, bv_j)$).
2) Based on the transferred weight vectors bw_k generate the initial rules,

$$IF\ (\bigwedge_{i=1,\ldots,N_H} LO\ F_{IHj}^{(k)})\ THEN\ CLS_k, \tag{31}$$

such that LO is void if $bw_{jk} = 1$ and LO is $\neg$ if $bw_{jk} = -1$. Note that if $bw_{jk} = 0$, the corresponding $F_{IH_j}^{(k)}$ is eliminated.

3) Expand each $F_{IH_j}^{(k)}$ in (31) according to the trinary vectors bv_j into the form,

$$F_{IH_j}^{(k)} \rightsquigarrow \bigwedge_{i=1,\ldots,N_I} x_i^{(k)} \mathcal{E} < x_{iS}^{(k)}, x_{iE}^{(k)} >, \tag{32}$$

such that $\mathcal{E}$ is $\in$ if $bv_{ij} = 1$ and $\mathcal{E}$ is $\notin$ if $bv_{ij} = -1$. Again, if $bv_{ij} = 0$ the corresponding $x_i^{(k)}$ is eliminated.

4) Simplify the expanded rules by merging the same statements and eliminating contradictions. Then present the rules of the form,

$$IF\ (\bigwedge_{i=1,\ldots,N_I} LO x_i^{(k)} \mathcal{E} < x_{iS}^{(k)}, x_{iE}^{(k)} >)\ THEN\ CLS_k. \tag{33}$$

5) Find appropriate intervals $< x_{iS}^{(k)}, x_{iE}^{(k)} >$ and substitute them into (33). Finally, generate the resulting rules.

As it can be seen in step 4) the above described methodology for rule extraction contains mechanism of optimizing the rules. This allows obtaining relatively simple rules even if the network's structure was overdetermined for a given classification task. In other words, the structural redundancies of a neural network can partially be eliminated when finalizing the rules. This feature has an important practical value since the network's structures may not always be optimal for a given classification task. The simplification mechanism will practically be demonstrated in the next section.

Important part is also finding the appropriate intervals $< x_{iS}^{(k)}, x_{iE}^{(k)} >$ in the step 5). If the conditions of the theorem are satisfied and rule includes the term $x_i^{(k)} \in < x_{iS}^{(k)}, x_{iE}^{(k)} >$, then the intervals can be obtained directly by tracing the min/max values of input coordinates for given class CLS_k. If the rule contains the term $x_i^{(k)} \notin < x_{iS}^{(k)}, x_{iE}^{(k)} >$ and the conditions of the theorem are satisfied, again the intervals are obtained directly by eliminating the intervals of input coordinates for classes CLS_j, $j \neq k$. However, the functionality of the methodology extends even to the cases where the conditions of the theorem do not strictly hold. Then the step 5) can be slightly more complicated. The approaches for obtaining the intervals (or membership functions) can vary [59], [60]. From the practical experience we can recommend the method of dichotomizing the boundaries for conflict patterns.

7.1 Simulations

The effectiveness of the methodology described in the previous section is demonstrated on the IRIS data set [49]. First, the case of rule extraction for optimized structure of a network is presented and in the second case the rule simplification mechanism is demonstrated on the overdetermined network structure.

Structure modifying training techniques play an important role in rule extraction approaches. Essentially, they lead to simpler initial rules (31) after the first three steps in the proposed methodology. In the first experimental task a structure modifying training technique based on performance measures was used [31]. Back propagation learning was terminated when the expected mean square error was less than 0.057. Resulting network, after transforming the original weight vectors into trinary vectors, had 3-2-4 structure. The transformation of weight vectors was as follows.

$$\begin{aligned} w_1 &= [0,\ 2.72] \rightarrow [0,\ 1] \\ w_2 &= [0.51,\ -0.94] \rightarrow [1,\ -1] \\ w_3 &= [0.65,\ -1.74] \rightarrow [1,\ -1] \\ v_1 &= [0,\ 0,\ 2.15,\ 0] \rightarrow [0,\ 0,\ 1,\ 0] \\ v_2 &= [0,\ 0,\ 0,\ -1.7] \rightarrow [0,\ 0,\ 0,\ -1] \end{aligned}$$

According to the proposed rule extraction methodology, from the network's structure the following rules immediately imply.

$$IF\ (x_4^{(1)} \notin < x_{4S}^{(1)}, x_{4E}^{(1)} >)\ THEN\ CLS_1 \tag{34}$$

$$IF\ (x_3^{(2)} \in < x_{3S}^{(2)}, x_{3E}^{(2)} > \wedge\ x_4^{(2)} \in < x_{4S}^{(2)}, x_{4E}^{(2)} >)\ THEN\ CLS_2 \tag{35}$$

$$IF\ (x_3^{(3)} \in < x_{3S}^{(3)}, x_{3E}^{(3)} > \wedge\ x_4^{(3)} \in < x_{4S}^{(3)}, x_{4E}^{(3)} >)\ THEN\ CLS_3 \tag{36}$$

Since the above rules cannot be further simplified the next step is only to determine the appropriate intervals. By detecting the min/max values of input coordinates for different classes the following intervals are obtained.

$$x_4^{(1)} \notin <1.0,\ 2.5>;\ x_3^{(2)} \in <3.0,\ 5.1>;\ x_4^{(2)} \in <1.0,\ 1.8>;$$

$$x_3^{(3)} \in <4.5,\ 6.9>;\ x_4^{(3)} \in <1.4,\ 2.5>$$

Inserting these intervals into the rules (34), (35), (36) leads to 94.6 % correct classification (142 correctly classified patterns out of 150). Dichotomizing the boundary of the input coordinate x_4 for conflict samples results in modification of $x_4^{(2)}$ and $x_4^{(3)}$ intervals: $x_4^{(2)} \in <1.0,\ 1.7>$, $x_4^{(3)} \in <1.35,\ 2.5>$. Substituting these boundaries into (35) and (36) gives 97.3 % correct classification (146 correctly classified exemplars out of 150).

The second simulation example demonstrates the rules simplification mechanism. An overdetermined network with structure 4-3-3 was trained using back propagation with constant learning rate 0.7 until the expected mean square error was less

than 0.057. The following weight vectors and their codified trinary vectors were obtained.

$$w_1 = [0.02, -0.007, 1.003] \rightarrow [0, 0, 1]$$
$$w_2 = [1.1, 0.71, -1.28] \rightarrow [1, 1, -1]$$
$$w_3 = [-1.31, 1.03, 1.123] \rightarrow [-1, 1, 1]$$
$$v_1 = [1.69, 2.0, -2.68, -3.29] \rightarrow [1, 1, -1, -1]$$
$$v_2 = [-0.29, -0.46, 0.94, 0.81] \rightarrow [0, 0, 1, 1]$$
$$v_3 = [0.73, 1.68, -3.36, -1.85] \rightarrow [1, 1, -1, -1]$$

It can clearly be seen that the connectivity of the output units O_2 and O_3 to the hidden units H_1 and H_3 leads to logical contradictions in derived rules. Then by principle of eliminating the contradictions and redundant 0-connections the simplified structure is obtained. The simplified structure leads to the same rules for classes CLS_2 and CLS_3 as (35) and (36). The only rule which is different and slightly more complicated is the rule for class CLS_1.

$$IF\ (\bigwedge_{i=1,\,2} x_i^{(1)} \in < x_{iS}^{(1)}, x_{iE}^{(1)} > \wedge \bigwedge_{i=3,\,4} x_i^{(1)} \notin < x_{iS}^{(1)}, x_{iE}^{(1)} >)\ THEN\ CLS_1 \qquad (37)$$

By substituting the same intervals as in the previous case for the rules classifying the classes CLS_2, CLS_3, and the intervals: $x_1^{(1)} \in <4.3, 5.8>$, $x_2^{(1)} \in <2.3, 4.4>$, $x_3^{(1)} \notin <3.0, 6.0>$, $x_4^{(1)} \notin <1.0, 2.5>$, into the rule (37), it is again obtained 97.3 % correct classification.

Note that in this case the network's structure was highly redundant. The rule simplification mechanism was able to eliminate more than 57 % of unnecessary rule terms (or connections) which substantially clarified the final rules.

8. Conclusions

The article introduced a neural network based IAS that incorporates all the essential features of adaptability required by wide range of communication technologies. Novel and theoretically consistent approach allowed effective operation and dynamic interlink of modules that have been formerly treated as separate subdomains. The presented IAS is able to appropriately select learning instances, adapt its parameters and structure. Moreover, the system includes a rule extraction module that could serve as a logical interface between IAS and expert systems. IAS utilizes first order optimization techniques with superlinear convergence rates. First order optimization is sufficiently fast and computationally inexpensive - with linear computational complexities. Thanks to the speed and computational inexpensiveness of adaptable techniques the system can adapt fast even on large number of training data. Additional advantage of the presented concept is that adaptability at parametric, structural, and interface levels is simultaneous and dynamic. The kernel of the system, that is neural network, can automatically at each iteration of adaptation select different number of training exemplars, adjust the learning parameters such as learning rate and/or momentum term, and also appropriately alter the structure in order to reach the optimum learning performance. After completing the adaptation, the system is able to interpret the learned task by logical formalism such as crisp rules. This feature may have indispensable value in expert system applications.

This research was partially funded by the Hori Foundation for Promotion of Information Sciences. The authors would like to thank Dr. Naohiro Toda of Aichi Prefectural University for his valuable comments. Requests for reprints should be addressed to Professor Shiro USUI, Department of Information and Computer Sciences, Toyohashi University of Technology, Hibarigaoka, Toyohashi 441-8580, Japan.

Reviewed by: V. Olej, R. Jarina

Nomenclature

N_I	number of input units
N_0	number of output units
N_H	number of hidden units
N_F	number of free parameters of network
N_P	cardinality of training set
T	training set
x	input vector
y	output vector
E	objective (or error) function for a neural network
F	mapping of MLP network
F_{HO}	hidden-to-output submapping
F_{IH}	input-to-hidden mapping
f_j	nonlinear sigmoidal transfer function of the j-th hidden unit
$\Re$	real space
$\Re^{N_I}$	N_I dimensional real space (input space)
$\Re^{N_H}$	N_H dimensional real space (hidden space)
$\Re^{N_O}$	N_O dimensional real space (output space)
w_{kl}	hidden-to-output weight connection, from the k-th hidden unit to the l-th output unit, or input-to-output weight connection, from the k-th input unit to the l-th output unit
w	vector of hidden/input-to-output weights
bw	binary/trinary hidden/input-to-output weight vector
v_{ij}	input-to-hidden weight connection, from the i-th input unit to the j-th hidden unit
v	weight vector of input-to-hidden weights
bv	binary/trinary input-to-hidden weight vector
Θ_h	threshold weight vector of hidden units
Θ_O	threshold weight vector of output units
u	set of free parameters of neural network
$u^{(0)}$	set of free initial parameters of neural network
u_l	the l-th free parameter of a neural network
$\alpha^{(n)}$	value of the learning rate at the n-th iteration
$\beta^{(n)}$	value of the momentum term at the n-th iteration
ρ	spectral radius estimate

Literatúra - References

[1] SHANNON, C. E., WEAVER, W.: The Mathematical Theory of Communications. University of Illinois Press, University of Illinois, 1949.
[2] ARBIB, M. A. (Editor): The Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, Massachusetts, 1995.
[3] BAUM, E., HAUSSLER, D.: What size of network gives valid generalization. Neural Computation, 1(1):151–160, 1989.
[4] HWANG, J., CHOI, J. J., SEHO, O., MARKS II, R. J.: Query learning based on boundary search and gradient computation of trained multilayer perceptrons. In Proceedings of IJCNN'90, pp. 57–62, San Diego, 1990.
[5] BAUM, E. B.: Neural net algorithm that learn in polynomial time for examples and queries. IEEE Trans. on Neural Networks, 2(1):5–19, 1991.
[6] R. Battiti. Using mutual information for selecting features in supervised neural net learning. IEEE Trans. on Neural Networks, 5(4):537–550, 1994.
[7] CACHIN, C.: Pedagogical pattern selection strategies. Neural Networks, 7(1):175–181, 1994.
[8] MUNRO, P. W.: Repeat until bored: A pattern selection strategy. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, Advances in Neural Information Processing Systems 4 (Denver), pp. 1001–1008, San Mateo, 1992. Morgan Kaufmann.
[9] FLETCHER, K.: Practical Methods of Optimization. John Wiley & Sons, Essex, 1987.
[10] WOLFE, P. Convergent conditions for ascent methods. SIAM Review, 11:226–235, 1969.
[11] POWELL, M. J. D.: A view of unconstrained optimization. In L. C. W. Dixon, editor, Optimization in Action, London, 1976. Academic Press.
[12] AL-BAALI, M., FLETCHER, R.: An ecient line search for nonlinear least squares. Journal of Optimization Theory and Application, 48(3):359–377, 1986.
[13] JACOBS, R. A.: Increasing rates of convergence through learning rate adaptation. Neural Networks, 1:295–307, 1988.
[14] T. P. Vogl, J. K. Manglis, A. K. Rigler, T. W. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. Biological Cybernetics, 59:257–263, 1988.
[15] PFLUG, Ch. G.: Non-asymptotic confidence bounds for stochastic approximation algorithms. Mathematic, 110:297–314, 1990.
[16] TOLLENAERE, T., SuperSAB: Fast adaptive back propagation with good scaling properties. Neural Networks, 3:561–573, 1990.
[17] DARKEN, C., MOODY, J.: Towards faster stochastic gradient search. In J. E. Moody, S. J. Hason, and R. P. Lipmann, editors, Proceedings of the Neural Information Processing Systems 4 (Denver), pp. 1009–1016, San Mateo, 1992. Morgan Kaufmann.
[18] OCHIAI, K., TODA, N., USUI, S.: Kick-Out learning algorithm to reduce the oscillation of weights. Neural Networks, 7(5):797–807, 1994.
[19] PERANTONIS, S. J., KARRAS, D. A.: An efficient constrained learning algorithm with momentum acceleration. Neural Networks, 8(2):237–249, 1995.
[20] BECKER, S., LEE CUN, Y.: Improving the convergence of back-propagation learning with second order methods. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, Proceedings of The 1988 Connectionist Models Summer School (Pittsburgh), pp. 62–72, N.Y., 1989. Wiley.
[21] BISHOP, C.: Exact calculation of the Hessian matrix for the multilayer perceptron. Neural Computation, 4(4):494–501, 1992.
[22] YU, X., LOH, N. K., MILLER, W. C.: A new acceleration technique for the backpropagation algorithm. In Proceedings of The IEEE International Conference on Neural Networks, pp. 1157–1161, San Francisco, 1993.
[23] YU, X., CHEN, G., CHENG, S.: Dynamic learning rate optimization of the backpropagation algorithm. IEEE Transactions on Neural Networks, 6(3):669–677, 1995.
[24] HINTON, G. E.: Connectionist learning procedures. Technical Report CMU-CS-87-115, Carnegie-Mellon University, 1987.
[25] WEIGEND, S. A., RUMELHART, D. E., and HUBERMANA, B. A.: Generalization by weight elimination with application to forecasting. In R. P. Lippman, J. E. Moody, and D. S. Touretzky, editors, Advances in Neural Information Processing Systems 3, pp. 875–882, San Mateo, 1991. Morgan Kaufmann.
[26] LECUN, Y., DENKER, J. S., SOLLA, S. A.: Optimal brain damage. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 2, pp. 598–605, San Mateo, 1990. Morgan Kaufmann.
[27] HASSIBI, B., STORK, D. G., WOLF, G. J.: Optimal brain surgeon and general network pruning. In IEEE International Conference on Neural Networks, pp. 293–299, San Francisco, 1993.
[28] CIBAS, T., SOULIÉ, F. F., GALLINARI, P., RANDYS, S.: Variable selection with neural networks. Neurocomputing, 12:223–248, 1996.
[29] GÉCZY, P., USUI, S.: Learning performance measures for MLP networks. In Proceedings of ICNN'97, pp. 1845–1850, Houston, 1997.
[30] GÉCZY, P., USUI, S.: Effects of structural adjustments on the estimate of spectral radius of error matrices. In Proceedings of ICNN'97, pp. 1862–1867, Houston, 1997.
[31] GÉCZY, P., USUI, S.: Effects of structural modifications of a network on Jacobean and error matrices. Submitted to Neural Networks, March 1997.
[32] TOWELL, G., SHAVLIK, J. W.: Extracting refined rules from knowledge-based neural networks. Machine Learning, 13:71–101, 1993.

[33] ANDREWS, R., DIEDERICH, J., TICKLE, A. B.: A survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems, 8:373–389, 1995.
[34] KASABOV, N.: Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. Fuzzy Sets and Systems, 2:135–149, 1996.
[35] FU, L.: Rule generation from neural networks. IEEE Transactions on SMC, 24:1114–1124, 1994.
[36] GÉCZY, P., USUI, S.: Rule extraction from trained artificial neural networks. In Proceedings of ICONIP'97, pp.835–838, Dunedin, 1997.
[37] GÉCZY, P., USUI, S.: Fuzzy rule acquisition from trained artificial neural networks. Journal of Advanced Computational Intelligence (accepted), March 1998.
[38] GÉCZY, P., USUI, S.: Rule extraction from trained artificial neural networks. BEHAVIORMETRIKA, 26(1):89–106, 1999.
[39] GÉCZY, P., USUI, S.: Knowledge acquisition from networks of abstrast bio-neurons. In Proceedings of ICONIP'99, pp. 610–615, Perth, 1999.
[40] HORNIK, K.: Multilayer feedforward networks are universal approximators. Neural Networks, 2:359–366, 1989.
[41] MHASKAR, H. N.: Neural networks for optimal approximation of smooth and analytic functions. Neural Computation, 8:164–177, 1995.
[42] GÉCZY, P., USUI, S.: A novel dynamic sample selection algorithm for accelerated learning. Technical Report NC97-03, IEICE, pp. 189–196, March 1997.
[43] GÉCZY, P., USUI, S.: Sample selection algorithm utilizing Lipschitz continuity condition. In Proceedings of JNNS'97, pp.190–191, Kanazawa, 1997.
[44] GÉCZY, P., USUI, S.: Dynamic sample selection: Theory. IEICE Transactions on Fundamentals, E81-A(9):1931–1939, 1998.
[45] GÉCZY, P., USUI, S.: Dynamic sample selection: Implementation. IEICE Transactions on Fundamentals, E81-A(9):1940–1947, 1998.
[46] GÉCZY, P., USUI, S.: Deterministic approach to dynamic sample selection. In Proceedings of ICONIP'98, pp. 1612–1615, Kitakyushu, 1998.
[47] ARMIJO, L.: Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of Mathematics, 16(1):1–3, 1966.
[48] CENDROWSKA, J., Prism: An algorithm for inducing modular rules. International Journal of Man-Machine Studies, 27:349–370, 1987.
[49] FISHER, R. A.: The use of multiple measurements in taxonomic problems. Annual Eugenics,7(II):179–188, 1936.
[50] DASARATHY, B. V.: Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(1):67–71, 1980.
[51] GÉCZY, P., USUI, S.: Fast back-propagation with automatically adjustable learning rate. Technical Report NC97-61, IEICE, pp. 47–54, December 1997.
[52] GÉCZY, P., USUI, S.: Superlinear and automatically adaptable conjugate gradient training algorithm. Technical Report NC97-149, IEICE, pp. 71–78, March 1998.
[53] GÉCZY, P., USUI, S.: On design of superlinear first order automatic machine learning techniques. In Proceedings of WCCI'98, pp.51–56, Anchorage, 1998.
[54] GÉCZY, P., USUI, S.: Novel first order optimization classification framework. IEICE Transactions on Fundamentals, Submitted(June), 1999.
[55] GÉCZY, P., USUI, S.: Superlinear conjugate gradient method with adaptable step length and constant momentum term. IEICE Transactions on Fundamentals, Submitted(June), 1999.
[56] GÉCZY, P., USUI, S.: Universal superlinear learning algorithm design. IEEE Transactions on Neural Networks, Submitted(February), 1999.
[57] FLETCHER, R., REEVES, C. M.: Function minimization by conjugate gradients. Comput. Journal, 7:149–154, 1964.
[58] WNEK, J., MICHALSKI, R. S.: Comparing symbolic and subsymbolic learning: Three studies. In R. S. Michalski and G. Tecuci, editors, Machine Learning: A Multistrategy Approach, volume 4, San Mateo, 1993. Morgan Kaufmann.
[59] ALEFELD, G., HERZBERGER, J.: Introduction to Interval Computations. Academic Press, New York, 1983.
[60] DUDA, R. O., HART, P E.: Pattern Classification and Scene Analysis. John Wiley & Sons, 1973.