
PERGAMON

Neural Networks 14 (2001) 551–573

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

Life-long learning Cell Structures—continuously learning without catastrophic interference

Fred H. Hamker

California Institute of Technology, Division of Biology 139-74, Pasadena, CA 91125, USA

Received 18 October 1999; accepted 5 January 2001

Abstract

As an extension of on-line learning, life-long learning challenges a system which is exposed to patterns from a changing environment during its entire lifespan. An autonomous system should not only integrate new knowledge on-line into its memory, but also preserve the knowledge learned by previous interactions. Thus, life-long learning implies the fundamental Stability–Plasticity Dilemma, which addresses the problem of learning new patterns without forgetting old prototype patterns. We propose an extension to the known Cell Structures, growing Radial Basis Function-like networks, that enables them to learn their number of nodes needed to solve a current task and to dynamically adapt the learning rate of each node separately. As shown in several simulations, the resulting Life-long Learning Cell Structures posses the major characteristics needed to cope with the Stability–Plasticity Dilemma.

Keywords: Life-long learning; Continuously learning; Incremental learning; Stability–Plasticity Dilemma; Catastrophic interference; Radial Basis Function; Cell Structures

1. Introduction

If we intend to bridge the gap between the learning abilities of humans and machines, at least on an abstract level of description, then we need to consider which circumstances allow a sequential acquisition of knowledge. In humans, two different systems, the hippocampus and the neocortex, seem to interact to achieve this capability (McClelland, McNaughton, & O'Reilly, 1995). Remarkably, the hippocampus retains its ability to generate neurons throughout life (Eriksson, Perfilieva, Bjork-Eriksson, Alborn, Nordborg, Peterson et al., 1998), especially in a stimulating environment, indicating a possibility for network growth. Functionally, attention and resonance seem to be of fundamental importance (Grossberg & Merrill, 1996). The idea is to select a subset of neurons by an attentional mechanism, ideally those which match the pattern best. Without destroying other prototype patterns, only the weights of these neurons are adapted. No convincing network exists yet that overcomes the catastrophic interference in sequential learning tasks and shows a good approximation as well. Although this contribution addresses learning predominantly from a technical point of view, similar mechanisms might exist in biological organisms. Specifically, we suggest that the stability/plasticity can be modulated by adaptive learning and insertion rates based on the observation of wrong responses.

1.1. The problem: learning in a non-stationary environment

A shortcoming from the biological, as well as the technical, point of view originates from the artificial separation of a lifespan into a learning and recognition phase. While this approach is possible for systems that operate in a fixed environment, it fails if the environment changes. To circumvent costly retraining, recent research in on-line learning focuses on adaptive learning rates to follow a non-stationary input distribution. Incremental learning addresses the ability of repeatedly training a network with new data, without destroying the old prototype patterns. Life-long learning, also termed continuous learning, emphasizes learning through the entire lifespan of a system (as opposed to the cases, where the term life-long learning is also used to address task independent learning).

On account of noise and other influences in learning an open data-set, possible discrete overlaps of decision areas turn into continuous overlaps and non-separable areas emerge. Furthermore, decision boundaries may change over time. In contrast to only adapting to a changing environment, life-long learning suggests preserving

0893-6080/01/$ - see front matter
PII: S0893-6080(01)00018-1

previously learned knowledge if it does not contradict the current task. This demand immediately raises the Stability–Plasticity Dilemma (Grossberg, 1988). We have to face the problem that learning in artificial neural networks inevitably implies forgetting. Later input patterns tend to wash out prior knowledge. While a gradual interference is unavoidable, the sudden and complete erasure of previously well learned patterns—a catastrophic interference—severely limits life-long learning. A purely stable network is unable to absorb new knowledge from its interactions, whereas a purely plastic network cannot preserve its knowledge. Thus, the issue of life-long learning addresses both. Is the network flexible enough to learn new patterns and can the network preserve old prototype patterns?

1.2. Previous research

Networks with a global or distributed representation of knowledge, like a Multi-Layer-Perceptron trained by error back-propagation (Rumelhart, Hinton, & Williams, 1986) and Cascade Correlation (Fahlman & Lebiere, 1990) suffer from the disadvantage that changing only one weight affects nearly all patterns stored in the network. Several modifications have been proposed, especially in reducing the representational overlap (see French, 1999, for an overview), but most of these altered networks show an overall performance decrease.

ART networks (e.g. Grossberg, 1976) circumscribe the dilemma by introducing a similarity criterion (vigilance), which allows learning only if the pattern sufficiently matches the stored prototype. The vigilance parameter is not necessarily restricted to remain constant over time. If the vigilance criterion is increased, only patterns sufficiently similar to the prototypes will provoke learning. If the criterion is not fulfilled, the pattern is represented by a newly assigned node, without a harmful interference with any learning previously accomplished. Although this similarity criterion works well in unsupervised learning, error-driven learning can result in a catastrophic allocation of new nodes. This occurs in overlapping decision regions. The Inter-ART-Reset in ARTMAP (Carpenter, Grossberg, & Reynolds, 1991) initiates the allocation of a new node, because it cannot reliably detect the appropriate prototype.

To combine the advantages of Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds, & Rosen, 1992) and the Probabilistic Neural Networks (Specht, 1990) for incremental learning, a hybrid network was proposed (Lim & Harrison, 1997), which achieved significantly better results than a regular Fuzzy ARTMAP on Gaussian source separation and on a noisy waveform recognition task. However, the authors investigated their network only on on-line learning tasks and not on the hard problems of life-long learning, such as continuous overlaps.

Another type of a local representation utilized successfully are Radial Basis Function (RBF) networks (Broomhead & Lowe, 1988; Moody & Darken, 1988). Nevertheless, those have a fixed number of nodes, which has to be determined by the designer. In a local representation, the best way to tackle the Stability–Plasticity Dilemma is to allocate new nodes if the present ones are not sufficient.

There have been numerous attempts to insert new nodes in RBF networks during learning (Berthold & Diamond, 1995; Chen, Thomas, & Nixon, 1994; Karayiannis & Mi, 1997; Obradovic, 1996; Platt, 1991; Roy, Govil, & Miranda, 1997; Shadafan & Niranjan, 1994; Whitehead & Choate, 1994; Yingwei, Sundararajan, & Saratchandran, 1998), but no criteria were defined that enable learning throughout the entire lifespan.

A promising strategy is the usage of a local error based insertion criterion (Fritzke, 1992, 1994) which became famous as a general criterion to determine the location where a RBF network should grow (Karayiannis & Mi, 1997). According to this concept, the nodes in a representation layer compete for determining the node with the highest similarity to the pattern. One might recognize this competition as comparable to the previously mentioned aim of additional selection. A counter linked with the winner is increased by the error of the network. On time average, nodes with high errors serve as a criterion to insert a new node. Thus, new nodes are inserted in areas of the input space where it is statistically indicated.

Two almost identical algorithms based on this criterion have been proposed, the Growing Neural Gas (Fritzke, 1995) and the Dynamic Cell Structures (Bruske & Sommer, 1995). They combine the idea of vector quantization with a continuous neighborhood adaptation. Both refer to the Growing Cell Structures (Fritzke, 1992, 1994), whose neighborhood relation is restricted to a fixed topology dimension.

Growing Cell Structures (GCS), Growing Neural Gas (GNG) and Dynamic Cell Structures (DCS) can be equipped with different learning rules and activation functions, such as RBF, and have been shown to achieve excellent results in different tasks (Bruske, Hansen, Riehnie, & Sommer, 1996; Bruske, Ahrns, & Sommer, 1998; Fritzke, 1994; Heinke & Hamker, 1998; Hamker, Paetz, Thöne, Brause, & Hanisch, 2000). Because of their similarity, GCS, GNG and DCS are hereafter called Cell Structures.

The original formulation of the algorithm intends an exclusively unsupervised adaptation of the input weights (Bruske & Sommer, 1995; Fritzke, 1994, 1995) similar to Self-Organizing Maps (Kohonen, 1982) and Neural Gas (Martinetz & Schulten, 1991, 1994), but without a decay of learning parameters. In contrast to the Self-Organizing Maps, the Neural Gas learns by a soft-max rule, an extension to the standard K-means clustering. This rule ranks the preference vectors according to their distance to the input pattern. Enriched by a competitive Hebbian on-line learning rule for the connections between nodes, the Neural Gas is able to learn perfectly topology preserving mappings (Martinetz & Schulten, 1994).

The Cell Structures have the main advantage that they neither need a priori decision about the network size, nor

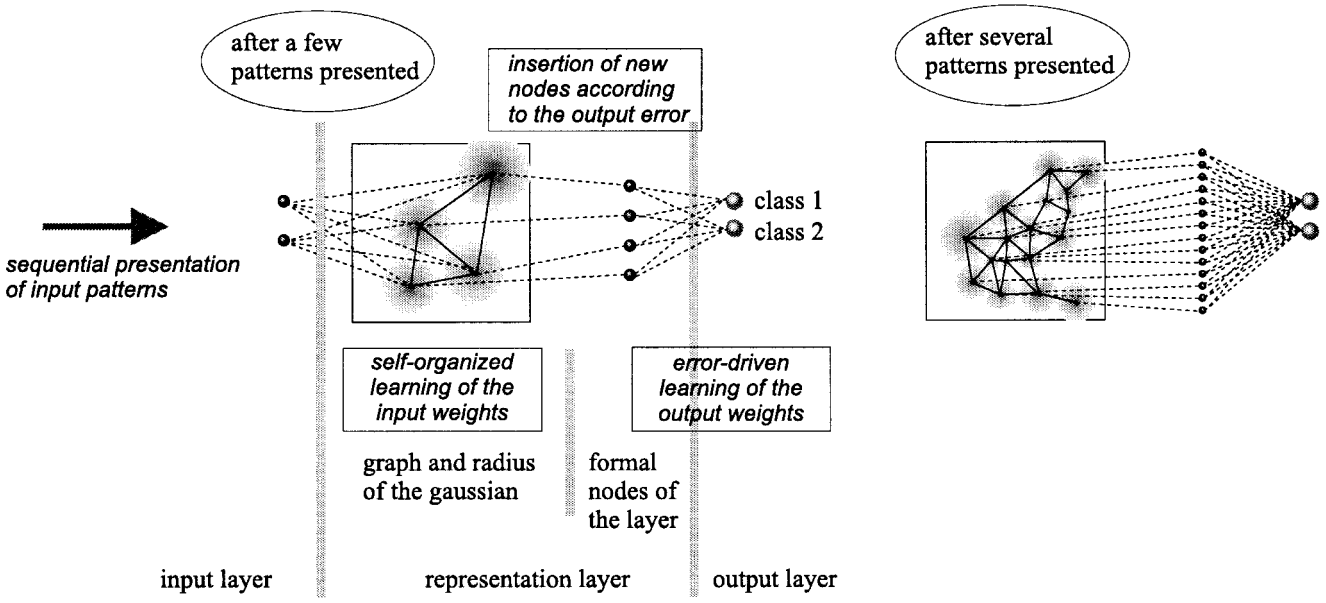


Fig. 1. On the left the layout of the Cell Structures is shown. The nodes in the representation layer adapt their weights according to the input pattern, usually based on an unsupervised learning rule. The graph is updated on-line. All nodes which share a common edge of a node are called neighbors of this node. The representation layer, composed of Gaussian activation functions, performs an adjustable, nonlinear transformation of the input pattern. The output layer then implements a separation into decision regions. In extension to the input and output weights adaptation the algorithm inserts new nodes in regions, which lead to high errors. Thus, beginning with a broad separation the network tries to fit even complex class boundaries in the data by inserting new nodes as illustrated on the right.

about the dimension of the graph, nor do they have to perform a ranking of the nodes in each adaptation step. They update their neighborhood relation on-line and utilize this information in each adaptation step. This cooperative training performs a similarity regularization, which generally improves training performance. Especially if the RBFs are large compared to the distance of the centers, the limited neighborhood adaptation of the Cell Structures shows a better generalization (Bruske, 1998). An output layer allows the separation of the activation in the representation layer into different classes. In this context, Cell Structures cluster the input space like RBF networks, but the nodes are organized within a graph in which the location of the centers and the connecting edges are updated on-line (Fig. 1).

In extension to the unsupervised learning of the input weights, an error-modulated learning (Ahrns, Bruske & Sommer, 1995) and a more sophisticated gradient-based learning (Bruske et al., 1996) have been suggested. Both introduce an error dependency for the adaptation of centers similar to the one utilized in various RBF algorithms (Karayiannis, 1999).

A utility-based removal (Fritzke, 1997) deletes nodes which are located in regions of a low input probability density. This criterion serves to follow a non-stationary input distribution and, therefore, counteracts the preservation of old prototype patterns.

So far, the major drawback of Cell Structures used in the context of life-long learning is their permanent increase in the number of nodes and in the drift of the centers to capture the input probability density in the unsupervised adaptation case and to reflect the error probability density in the supervised adaptation case (Hamker & Gross, 1997). Thresholds such as a maximal number of nodes predetermined by the user as well as an insertion criterion dependent on the overall error or on a quantization error are not appropriate, simply because appropriate figures for these criteria cannot be known in advance. Thus, the Cell Structures are only capable to follow an input probability density by using the utility-based removal (high plasticity) or to approximate a decision boundary by freezing the number of nodes and allowing only minor changes in the adaptation of the weights (high stability).

The catch is that these networks suffer even more from a catastrophic allocation of new nodes (Hamker & Gross, 1997), because they permanently insert new nodes at locations with high errors. Thus, we propose an extension of the Cell Structures in order to balance the stability and plasticity dynamically. The next section gives a brief overview of the problems of learning in a non-stationary environment and our suggested solutions.

1.3. A new approach: life-long learning Cell Structures

The previous section indicates that current networks either fail to meet the demand for stability or do not show enough plasticity while learning in a non-stationary environment. We emphasized that a local representation, an insertion of new nodes and an adaptive learning rate are key aspects to reduce the interference.

The insertion of new nodes is a useful contribution to the

plasticity aspect, but burdened with overfitting and catastrophic waste of resources. Incremental networks such as the Cell Structures cannot escape that fact without the ability to learn whether a further insertion of nodes is useful or not. Thus, we suggest a strategy that allows insertion only if a local error counter exceeds a local insertion threshold, which is permanently adapted to the present situation.

The next key aspect refers to the learning of the weights. There are many reasons why the learning rate should be adaptive. If the learning rate is held constant, the learner can never converge to the global optimum: a learning rate chosen too high will corrupt previously learned knowledge and disturb following stages, a learning rate chosen too low does not allow the learner to follow a changing environment. Since Cell Structures provide a local processing strategy, a uniform learning rate does not make sense. Each node should hold its individual adaptive learning rate. We suggest estimating the appropriate learning rate by two error counters with different time-constants. This method detects relevant, local changes in the input probability density. Irrelevant changes, which do not affect the error on the task, or changes in a different area of the input space do not lead to more plasticity. Both key aspects, the adaptive insertion and the adaptive learning parameters, are now discussed in more detail.

1.3.1. Adaptive insertion

A general problem in learning is the Bias–Variance Dilemma (Geman, Bienenstock, & Doursat, 1992), where the bias is a measure of the similarity of the average mapping function to the desired one and the variance describes the confidence of the mapping function concerning different input patterns. The conflict lies in minimizing the bias and avoiding a high variance, often termed as a good generalization.

Theoretically, RBF networks have general approximation capabilities (Park & Sandberg, 1993; Poggio & Girosi, 1990), but in practical applications the best solution is unknown and the optimal number of units in the representation layer cannot be found. In incremental neural networks, e.g. the Cell Structures, insertion is used to improve the mapping until a criterion is reached, e.g. a minimal overall error, a maximal number of nodes, or a low error on a validation data set.

In life-long learning tasks, growth is an important feature to decrease the error of the task and to adapt to changing environments while preserving old prototype patterns, but the mentioned criteria are not appropriate. Nevertheless, insertion must be stopped for two reasons: to prohibit a permanent increase in the number of nodes in overlapping decision areas, where the task cannot be solved, and to avoid overfitting. The learning of the insertion parameter can be explained by an insertion–evaluation cycle (Hamker, 1999). After a number of learning steps, the average error counter of a node is compared to the error at the moment of the last insertion. If the current error is greater or equal, the insertion was not successful and a local insertion threshold is increased. If the threshold reaches the average error, a further insertion at that location is not allowed. To permit exploration in the future, the threshold should be decreased by some criterion as explained in Section 2.2.

1.3.2. Adaptive learning rate

As a common procedure to minimize the bias in on-line learning, a stochastic gradient descent is used, in which the learning rate is decreased to zero (Robbins & Monro, 1951). Thus, the Stability–Plasticity Dilemma is interpreted in favor of the stability. In a changing environment, this approach turns into a conflict, as an annealed learning rate does not allow the weights to follow the changes fast enough. To overcome this problem, different methods of adapting the scale factor of the learning parameter have been proposed. Thus, the adjustment of the weight vector Δw depends on the old weight vector w and the current pattern x, scaled by an adaptive factor $\eta(t)$, which was termed by Amari (1967) as learning of a learning rule.

$$\Delta w = \eta(t) \cdot f(w, x) \tag{1}$$

Theoretical considerations of finding the optimal learning rate often assume a global scale factor for all weight-adjustments that depends on the past errors (Murata, Müller, Ziehe, & Amari, 1997; Sompolinsky, Barkai, & Seung, 1995; Freeman & Saad, 1997). If the error is large, the learning rate takes on a large value. If the error decreases to zero, the learning rate decreases to zero. A different approach almost independent from the type of neural network was proposed by Heskes and Kappen (1993). They derived the actual learning rate from minimizing a misadjustment, which is low with a small bias and a small variance. For practical purposes, they estimated the bias and the variance from the statistics of the weights by time averages over a period T, which has to be chosen according to the typical time scale of changes. Still, this measure does not take into account the differences of the distribution in the input space. For the Cell Structures, individual adaptive learning rates for each node are more suitable. To estimate a good level of the learning rate, we suggest using the ratio between two local error counters, each with a different time constant, because this guarantees an asymptotic decrease in the case of a local stationary distribution, and an increase if the changing environment leads to new local errors.

1.4. Outline

We have discussed the key issues of learning in a non-stationary environment and worked out methods to tackle the Stability–Plasticity Dilemma. The next section gives an overview of the developed algorithm by means of a pseudocode, followed by a complete description. By using artificial data, we illustrate the learning process and observe details, such as temporal snap shots of internal parameters, which helps to fully understand the learning behavior. The

evaluation of the Life-long Learning Cell Structures is achieved by a benchmark on real data with a stationary distribution and with a non-stationary distribution. The latter requires the definition of new criteria to evaluate the degree of stability and plasticity.

2. Life-long Learning Cell Structures

2.1. Overview

Learning in a non-stationary environment demands several extensions to the original algorithm. The main idea is to use local counters to evaluate the need of a local weight adaptation and of a local node insertion. The general course of events is given by the following pseudocode.

Initialization
Do randomly choose the input and output weights of a network with two nodes in the representation layer and connect both nodes with an edge. The age of each node is $Y = 1$. Since no pattern was presented yet, no errors occurred and the error counters of each node are $\tau_S = \tau_L = 0$. Any insertion should not be restricted in the beginning ($\tau_\vartheta = 0$). The inherited error is set to $\tau_I = 1$, which means that the first insertion initiated by each of those two nodes is evaluated as an improvement.
Repeat for each pattern:
Adaptation of the representation layer
Do locate the node b, which best matches the input pattern by applying a distance measure. Find also the second best node s.
Do determine the quality measure for learning B^L for b and its neighbors c separately, which is defined as the quotient of the short-term error counter τ_S and long-term error counter τ_L.
Do determine the individual input learning rate of b and of its neighbors c, which depends on each quality measure for learning B^L and on the age Y of each node. The younger the node and the larger τ_S over τ_L, the higher the learning rate. The cut-off value of learning is set by an input adaptation threshold ϑ_L^i.
Do move the input weights of the node b and its neighbors toward the presented pattern according to the individual learning rate.
Adaptation of the output layer
Do calculate the activation of all nodes in the representation layer by a RBF.
Do determine the individual output learning rate for all nodes in the representation layer. It is the same procedure as for the input learning rate, but a different adaptation threshold ϑ_L^o can be used.
Do adapt the output weights to match the desired output by applying the delta rule.
Insertion and deletion of nodes in the representation layer
If the algorithm was presented a sufficient number of patterns since the last insertion criterion was applied.
Check insertion
Do find the node q with the highest value of the insertion criterion. The insertion criterion depends on each quality measure for insertion B^I and on the age of each node. This quality measure is defined as the difference between the long-term error counter τ_L and the insertion threshold τ_ϑ, which is increased by an insertion tolerance. The younger the node and the larger τ_L as against τ_ϑ, the higher the insertion criterion.
If the insertion criterion of q is larger than zero
Insert a new node
Do find the node f among the neighbors of q, which shows the highest quality measure of insertion and insert a new node r between q and f. Connect the new node with q and f by new edges and delete the edge between q and f. Initialize all counters of r by an arithmetical average of the respective counters of q and f.
Evaluate the previous insertion
For q, f and r
If the current long-term error τ_L exceeds the inherited error τ_I lowered by the insertion tolerance
The last insertion was not successful
Do increase the insertion threshold τ_ϑ of the node.
End If
Error memorization
Do memorize the current long-term error τ_L in the inherited error τ_I.
End For
End If
Check similarity based deletion
Do find the node d with the lowest value of the deletion criterion. The smaller the distance of the input weights to the input weights of the neighbors as against the average distance of the input weights and the smaller the distance of the output weights to the output weights of the neighbors, the lower the deletion criterion.
If the deletion criterion of d is smaller than the threshold ϑ_{del}, which has to be determined by the user (further criteria can be added here)
Delete the node
Do remove the node d and update the graph.
End If
End If
Update the counters
Do update the long-term error τ_L and the short-term error τ_S for the winner b by calculating a moving average with particular time constants considering the current output error.
Do exponentially decrease the age of the winter b with a particular time constant.
Do exponentially decrease the insertion threshold τ_ϑ of

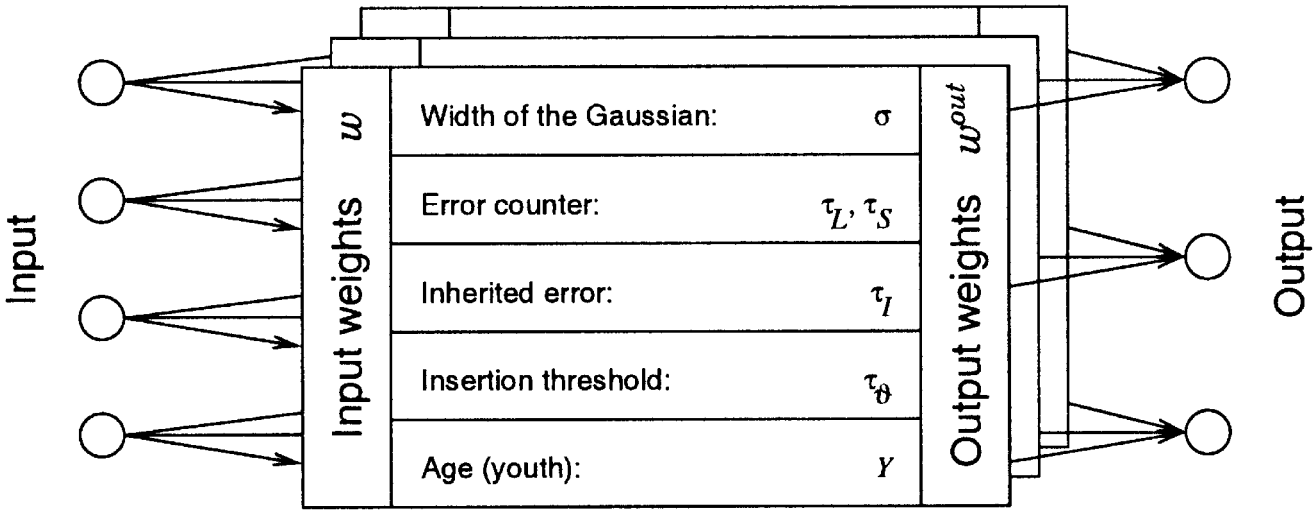


Fig. 2. A node of the Life-long Learning Cell Structures. Besides the width of the Gaussian, each node has error and age counters. In contrast to the inherited error, which remains fixed until the node is selected for insertion, the error counters are defined as moving averages according to their individual time constant.

the winner b depending on the quality measure for learning B^{L}. A decrease of the insertion threshold τ_ϑ only takes place if the current long-term error τ_{L} differs from the short-term error τ_{S}.

Update the edges of the graph

Do increase the age of all edges emanating from b.

Do set the age of the edge between b and s to zero. If no edge between b and s exists, create a new one.

Do remove all edges older than the threshold τ_{age}.

Do remove all nodes without any edge (only for consistency).

End Repeat

2.2. Complete algorithm for supervised classification

As an example, the algorithm is described for the task of supervised classification. In this learning scheme the network describes a special case of an incremental RBF network.

2.2.1. Structure

The representation layer of the Life-long Learning Cell Structures (LLCS) performs a vector quantization and consists of nodes or prototypes. The neighborhood relationship of the nodes is defined by an undirected graph G (Martinetz & Schulten, 1994). All edges that emanate from a node i determine its neighbors N_i. The age of each edge is continuously updated by a Hebbian adaptation rule. The total amount of nodes is denoted with n_N.

Variables of each node

Each node i has a few variables that regulate learning and the insertion of new nodes in the network (Fig. 2).

w_i n-dimensional weight vector in the input space

w_i^{out} m-dimensional weight vector in the output space

σ_i Width of the Gaussian. Extreme values of σ can be crucial to the performance. Good results are obtained by estimating the variance in the input data of each best-matching node or by simply averaging the length of all emanating edges:

$$\sigma_i = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i - w_j\| \tag{2}$$

$\tau_{\mathrm{S}i}$ Short-term error counter. The estimate of the average short-term error is updated according to the time constant T_{S}.

$\tau_{\mathrm{L}i}$ Long-term error counter. Similar to the short-term error counter, the variable estimates the average error, but considers a larger time constant T_{L}.

$\tau_{\mathrm{I}i}$ Inherited error. This variable serves as a memory for the error at the moment of insertion. It is updated at each insertion, but only for the affected nodes. The inherited error of a new node receives the long-term error τ_{L} of those two nodes, between which the new one is inserted. Both nodes memorize their present long-term error.

$\tau_{\vartheta i}$ Insertion threshold. An insertion is only allowed if the long-term error exceeds this local threshold. It is increased if a previous insertion was not successful. An exponential decrease according to the time constant T_ϑ depends on a relevant change in the input probability distribution.

Y_i Age of the node. It is decreased for the best-matching node according to the time constant T_Y.

Adaptation of the representation layer

- For all nodes i, calculate the Euclidian distance d_i of the input pattern $x \in \mathbb{R}^n$ to the weight vector $w_i \in \mathbb{R}^n$ and

locate the best-matching unit b and the second best s.

$$d_b = \min_{i \in G}(d_i); \qquad d_s = \min_{i \in G, i \neq b}(d_i); \qquad d_i = \|x - w_i\| \qquad \forall i \in G \tag{3}$$

- Determine the quality measure for learning B^{L} for the best-matching node b and its neighbors $c \in N_b$.

$$B^{\mathrm{L}}_{(b/c)} = \frac{\tau_{\mathrm{S}(b/c)} + 1}{\tau_{\mathrm{L}(b/c)} + 1} \qquad \forall c \in N_b \tag{4}$$

- Determine the input learning rate $\eta^{\mathrm{i}}_{(b/c)}$ of the best node b and its neighbors c from the quality measure for learning B^{L}, the age Y, the learning rate of the winner η_b respectively learning rate of the neighbors η_n and a pre-defined input adaptation threshold $\vartheta^{\mathrm{i}}_{\mathrm{L}}$.

$$\eta^{\mathrm{i}}_{(b/c)} = \begin{cases} 0 \text{ if } \alpha^{\mathrm{i}}_{(b/c)} < 0 \\ \eta_{(b/n)} \text{ if } \alpha^{\mathrm{i}}_{(b/c)} > 1 \text{ with } \alpha^{\mathrm{i}}_{(b/c)} = \dfrac{B^{\mathrm{L}}_{(b/c)}}{1 + \vartheta^{\mathrm{i}}_{\mathrm{L}}} + Y_{(b/c)} - 1 \\ \alpha^{\mathrm{i}}_{(b/c)} \eta_{(b/n)} \text{ else} \end{cases} \tag{5}$$

The learning rate η^{i} allows an adaptation of the weights either if the nodes are new or if temporal changes of the error occur. Within this adaptive phase, the network approximates the input probability density and does not account for the local distribution of the error (unsupervised rule). To approximate the error probability density, the learning rate must be extended by a gradient-based or error-modulated learning rule (Bruske, 1998).

- Increase match for b and its neighbors $c \in N_b$.

$$\Delta w_b = \eta^{\mathrm{i}}_b (x - w_b); \qquad \Delta w_c = \eta^{\mathrm{i}}_c (x - w_c) \qquad \forall c \in N_b \tag{6}$$

Adaptation of the output layer

- In case of the error-driven example discussed here, determine the Euclidian distance of the output $o \in \mathbb{R}^m$ to the target $\zeta \in \mathbb{R}^m$, when the input x is presented.

$$E_{\mathrm{task}}(x) = \|\zeta - o\| \tag{7}$$

- Calculate the activation of all nodes y_i with a Gaussian,

$$y_i = \mathrm{e}^{-\|x - w_i\|/\sigma_i^2} \qquad \forall i \in G \tag{8}$$

- Determine the local output learning rates η^{o} from the quality measure B^{L}, the age Y, the output adaptation rate η_o, and the output adaptation threshold $\vartheta^{\mathrm{o}}_{\mathrm{L}}$.

$$\eta^{\mathrm{o}}_i = \begin{cases} 0 \text{ if } \alpha^{\mathrm{o}}_i < 0 \\ \eta_o \text{ if } \alpha^{\mathrm{o}}_i > 1 \text{ with } \alpha^{\mathrm{o}}_i = \dfrac{B^{\mathrm{L}}_i}{1 + \vartheta^{\mathrm{o}}_{\mathrm{L}}} + Y_i - 1 \\ \alpha^{\mathrm{o}}_i \eta_o \text{ else} \end{cases} \qquad \forall i \in G \tag{9}$$

- Adapt the weights of the nodes j of the output layer.

$$\Delta w_{ji} = \eta^{\mathrm{o}}_i (\zeta_j - o_j) y_i \qquad \forall j \in \{1 \ldots m\},\ \forall i \in G \tag{10}$$

Insertion and deletion of nodes in the representation layer

- After each $T_{\mathrm{ins}} = \lambda \cdot n_N$ steps: determine the quality measure for insertion B^{I} considering the insertion tolerance ϑ_{ins}.

$$B^{\mathrm{I}}_i = \tau_{\mathrm{L}i} - \tau_{\vartheta i}(1 + \vartheta_{\mathrm{ins}}) \qquad \forall i \in G \tag{11}$$

- Find node q, which shows the maximal value of the insertion criterion K_{ins}, and search among its neighbors the node f, which shows the maximal value of the quality measure for insertion B^{I}. Insert a new node if the insertion criterion is satisfied.

$$0 < K_{\mathrm{ins},q} = \max_{i \in G}(K_{\mathrm{ins},i}); \qquad K_{\mathrm{ins},i} = B^{\mathrm{I}}_i - Y_i$$
$$\forall i \in G; \qquad B^{\mathrm{I}}_f = \max_{c \in N_q}(B^{\mathrm{I}}_c); \tag{12}$$

This criterion is only error based, which supports the acquisition of nodes in regions with high errors independent of the input probability density.

- If f and q exist: delete the edge between q and f, insert a new node r, and connect r with q and f. The weights w_r, w^{out}_r as well as the counters $\tau_{\mathrm{S}r}$, $\tau_{\mathrm{L}r}$, and $\tau_{\vartheta r}$ are determined by the arithmetical average of the corresponding weights and error counters of q and f.
  If the long-term error $\tau_{\mathrm{L}i}$ of q, f and r exceeds the inherited error $\tau_{\mathrm{I}i}$ lowered by an insertion tolerance ϑ_{ins}:

$$\tau_{\mathrm{L}i} \geq \tau_{\mathrm{I}i}(1 - \vartheta_{\mathrm{ins}}) \qquad \forall i \in \{q, f, r\} \tag{13}$$

the last insertion was not successful, and the insertion threshold is adapted.

$$\tau_{\vartheta i} := \tau_{\vartheta i} + \eta_\vartheta (\tau_{\mathrm{L}i} - \tau_{\vartheta i}(1 - \vartheta_{\mathrm{ins}}))$$
$$\forall i \in \{k | \tau_{\mathrm{L}k} \geq \tau_{\mathrm{I}k}(1 - \tau_{\mathrm{ins}}); q, f, r\} \tag{14}$$

Assign the inherited error $\tau_{\mathrm{I}i}$ of q, f and r to the present long-term error.

$$\tau_{\mathrm{I}i} = \tau_{\mathrm{L}i} \qquad \forall i \in \{q, r, f\} \tag{15}$$

If f and q do not exist, no insertion evaluation takes place.

- Check the deletion criteria considering a minimal age $\vartheta_{\text{del}Y}$, a sufficient stabilization $\vartheta_{\text{del}B^{\text{L}}}$ and the number of edges. Find the node d, whose criterion K is lower than the deletion threshold ϑ_{del}.

$$\vartheta_{\text{del}} > K_{\text{del},d} = \min_{i \in G}(K_{\text{del},i}) \wedge \|N_d\| \geq 2 \wedge Y_d < \vartheta_{\text{del}Y} \wedge B_d^{\text{L}} < \vartheta_{\text{del}B^{\text{L}}} \tag{16}$$

with

$$K_{\text{del},i} = \frac{\overline{\Delta w_i}}{\bar{l}} \overline{\Delta w_i^{\text{out}}} \qquad \forall i \in G \tag{17}$$

the local similarity of the input weights:

$$\overline{\Delta w_i} = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i - w_j\| \tag{18}$$

the average similarity of the input weights:

$$\bar{l} = \frac{1}{n_N} \sum_{j=1}^{n_N} \overline{\Delta w_j} \tag{19}$$

and the local similarity of the output weights:

$$\overline{\Delta w_i^{out}} = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i^{\text{out}} - w_j^{\text{out}}\| \tag{20}$$

Adaptation of the counters and edges of the nodes in the representation layer

- Update the long-term error counter τ_{L} and the short-term error counter τ_{S} for the winner b.

$$\tau_{(L/S)b} := \mathrm{e}^{-1/T_{(\text{L/S})}} \tau_{(\text{L/S})b} + \left(1 - \mathrm{e}^{-1/T_{(\text{L/S})}}\right) E_{\text{task}}(x) \tag{21}$$

- Decrease the age Y of the best-matching node b.

$$Y_b := \mathrm{e}^{-1/T_Y} Y_b \tag{22}$$

- Decrease of the insertion threshold $\tau_{\vartheta b}$, if the distribution of the error changes.

$$\tau_{\vartheta b} := (1 - \Lambda(\alpha_b))\mathrm{e}^{-1/T_\vartheta} \tau_{\vartheta b} \tag{23}$$

with

$$\alpha_b = \frac{1 + |B_b^{\text{L}} - 1|}{1 + \vartheta_{\text{L}}^{\text{i}}} - 1; \qquad \Lambda(x) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x > 1 \\ x \text{ else} \end{cases} \tag{24}$$

- Adapt the edges as follows:
  Increase the age of all edges emanating from b by one.
  Set the age of the edge between b and s to zero. If no edge between b and s exists, create a new one.
  Remove all edges older than ϑ_{age}
  Remove all nodes without any edge.

2.3. Parameter discussion

We assume the specification of several parameters for the algorithm. The major parameters that concern the insertion and deletion, i.e. the size of the network, are the learning rate η_ϑ of the adaptive insertion threshold, and the deletion threshold ϑ_{del}. The sensitivity to temporal changes of the environment is adjusted by the relation of the time constants for the short-term error and the long-term error $T_{\text{S}}/T_{\text{L}}$. We discuss this by means of a continuously moving class (Fig. 3). The network follows the non-stationary distribution and leaves a track. The remained nodes memorize the knowledge to separate between the classes even in those areas that were not visited any more (Fig. 3 right). The track turns out to be less marked if only a few nodes are inserted and if the weights are allowed to adapt strongly. Concerning the latter case, the time constants T_{S}, T_{L} and T_Y are decisive. A low T_{S} compared to T_{L} makes the network more sensitive to changes of the environment, but less lasting. Furthermore, a low time constant for the age of a node T_Y quickly reduces the preference of new nodes to highly change their weights. This will be discussed in detail by means of Fig. 4.

Concerning the insertion of nodes, the learning rate η_ϑ of

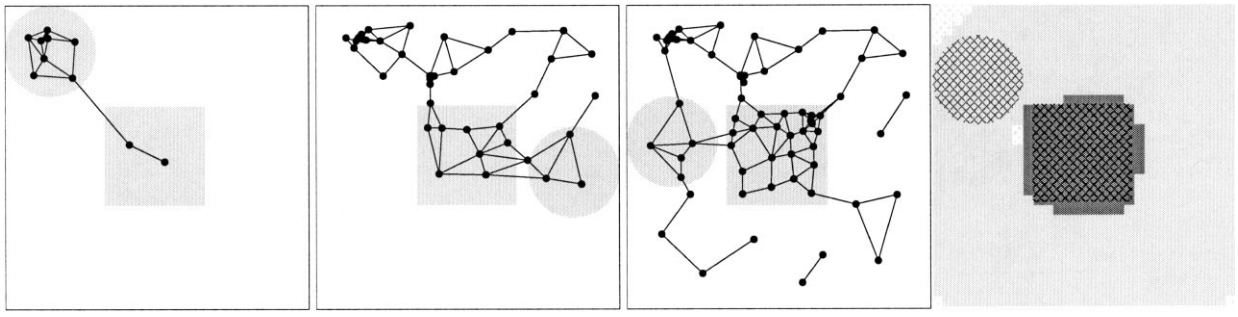


Fig. 3. Nodes and class decision (right) in a continuously changing environment. One class moves around another class. Patterns are only represented within the rectangular and the circular area. Simulation parameters: $\eta_b = 0.1$, $\eta_n = 0.01$, $\eta_o = 0.1$, $\eta_\vartheta = 0.1$, $T_{\text{S}} = 10$, $T_{\text{L}} = T_Y = T_\vartheta = 100$, $\lambda = 10$, $\vartheta_{\text{age}} = 50$, $\vartheta_{\text{L}}^{\text{i}} = 0.001$, $\vartheta_{\text{L}}^{\text{o}} = -0.05$, $\vartheta_{\text{ins}} = 0.01$, $\vartheta_{\text{del}} = 0.01$, $\vartheta_{\text{del}Y} = 0.01$, $\vartheta_{\text{del}B^{\text{L}}} = 0.01$.

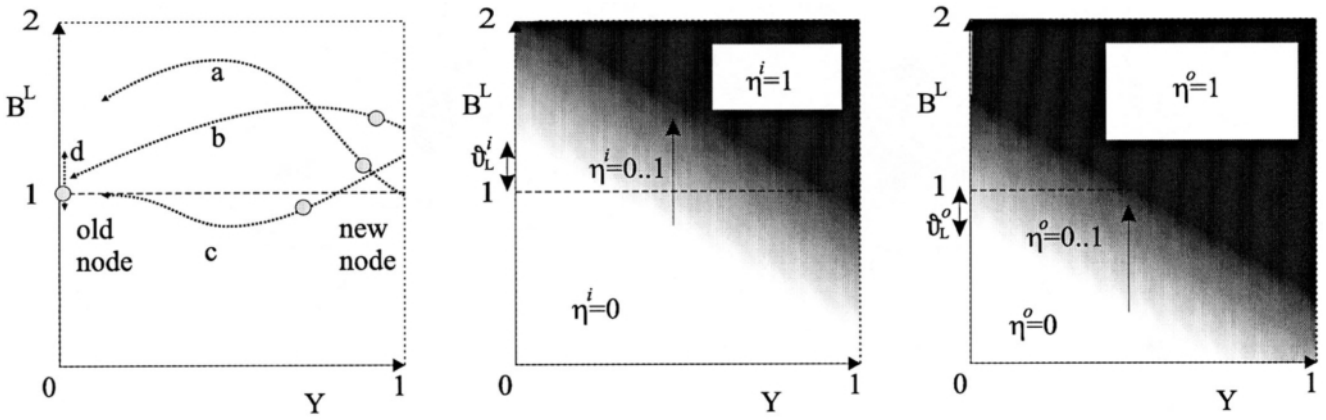


Fig. 4. The learning rate η in dependence of the quality measure for learning B^L, the age Y and the input/output adaptation threshold $\vartheta_L^{i/o}$. Left: typical states of the nodes during learning. Please see the text for a discussion of the different cases. Middle: learning of the centers and the influence of the user defined input adaptation threshold ϑ_L^i. The younger the node and the higher the quality measure for learning B^L, the higher the learning rate η^i. A positive value for ϑ_L^i prevents learning for old nodes, if the long-term error T_L is equal to the short-term error T_S. Right: learning of the output weights and the influence of the user defined output adaptation threshold ϑ_L^o. Usually, the value for ϑ_L^o is chosen to be negative, which allows the output weights to learn the decision boundary even when the input distribution is stationary.

the adaptive insertion threshold and a tolerance threshold for insertion ϑ_{ins} are most important. The tolerance threshold for insertion ϑ_{ins} defines the sensitivity to changes of the long-term error τ_L. A low value enhances insertion. However, for real data a too low threshold is undesirable, because natural variations of the long-term error may often result in unintentional insertions of nodes, especially in overlapping decision areas. To facilitate insertion in general, we recommend decreasing the learning rate of the adaptive insertion threshold η_ϑ. This results in a slower adaptation of the insertion threshold τ_ϑ to the value of the long-term error τ_L.

We would like to emphasize that the deletion of a node, since it has lost all its edges, is a very rare exception. Take the example of the continuously moving class (Fig. 3). Even when no pattern in the rectangular area was presented any more, the nodes were not removed, because only edge counters emanating from the winner are decreased. Only if a single node represents an area in which patterns no longer appear, is it deleted because of the decrease of the edge. If this should be prevented, consider using an asymmetric rule, which is based on directed edges (Bruske, 1998). Following this rule, the connecting edge is only deleted if the age of both directions falls below a threshold.

Looking back at the previous discussion on the insertion of nodes, note that an incremental neural network cannot assess in advance whether a further insertion reveals a subtle distribution or turns out to be a waste of resources. This is closely related to the Bias–Variance Dilemma. Bias means the average deviation between the output of the training pattern and the expected output. The variance indicates the sensitivity of the output concerning different patterns. An insertion increases the number of free parameters and improves the network performance on the current data, but might result in a loss of generalization. The proposed strategy to evaluate an insertion locally is a suitable criterion for simultaneously minimizing both bias (due to the ability of insertion) and variance (due to evaluation and, if applicable, the suppression of insertion). In the Life-long Learning Cell Structures the learning rate of the insertion threshold η_ϑ determines this generalization property. The larger the learning rate of the insertion threshold η_ϑ, the larger the effect of a wrong insertion and the less insertions are possible until the insertion threshold reaches the long-term error. This criterion discovers the decision boundaries between distinctly separated classes, but avoids a too low bias in areas with much overlap. Another criterion acts after an insertion and removes similar nodes. The larger ϑ_{del}, the earlier similar nodes will be deleted. Both parameters together allow the user to adjust the bias. Table 1 illustrates how to determine the generalization by choosing different values.

After some time, the parameters became very intuitive, because they are linked directly to the network's behavior. Important parameters are the learning rate of the adaptive insertion threshold η_ϑ, the deletion threshold ϑ_{del}, and the relation of the time constants of the short-term error, and the long-term error T_S/T_L. For most applications, the other parameters can be regarded as constants of the algorithm. We found the parameters for the experiments described in this article by trial and error. However, we would like to point out that a broad range of combinations are suitable for good results, as shown in Section 4. This insensitivity to parameter settings is a general feature of the local processing strategy in the Cell Structures as indicated by a benchmark (Heinke & Hamker, 1998).

2.4. Learning process

Learning of a node is illustrated by plotting its values for the quality measures for learning B^L and the age Y (Fig. 4). The amount of learning depends on its location within this

Table 1
Illustration of the quantization depending on the learning rate of the insertion threshold η_ϑ and the deletion threshold ϑ_{del} in an environment which shows three partly overlapping classes. One figure depicts the network and the areas with input data, marked by gray circles (the boundaries between the classes are not shown), the other figure represents the number of nodes (-·-) and the classification error on the training data (—) over time. Clearly visible by the error and the sparse graph of the network, larger values of η_ϑ lead to an extraction of the rudimentary class distribution. Lower values allow the network to detect details of the class distribution in the training data. With these parameters, several nodes are clumped nearby. But because some classes overlap, not even many more nodes can solve the task better. Thus, the deletion criterion keeps the number of nodes small, without affecting the performance on the training data badly. A value of $\vartheta_{\text{del}} < 0$ prohibits a deletion of nodes. Concluding, the learning rate of the insertion threshold η_ϑ and the deletion threshold ϑ_{del} allow the user to determine the desired generalization ability. Simulation parameters: $\eta_b = 0.1$, $\eta_n = 0.01$, $\eta_o = 0.15$, $T_{\text{S}} = 20$, $T_{\text{L}} = T_Y = T_\vartheta = 100$, $\lambda = 10$, $\vartheta_{\text{age}} = 50$, $\vartheta_{\text{L}}^{\text{i}} = 0.05$, $\vartheta_{\text{L}}^{\text{o}} = -0.05$, $\vartheta_{\text{ins}} = 0.2$, $\vartheta_{\text{del}Y} = 0.01$, $\vartheta_{\text{del}B^{\text{L}}} = 0.01$

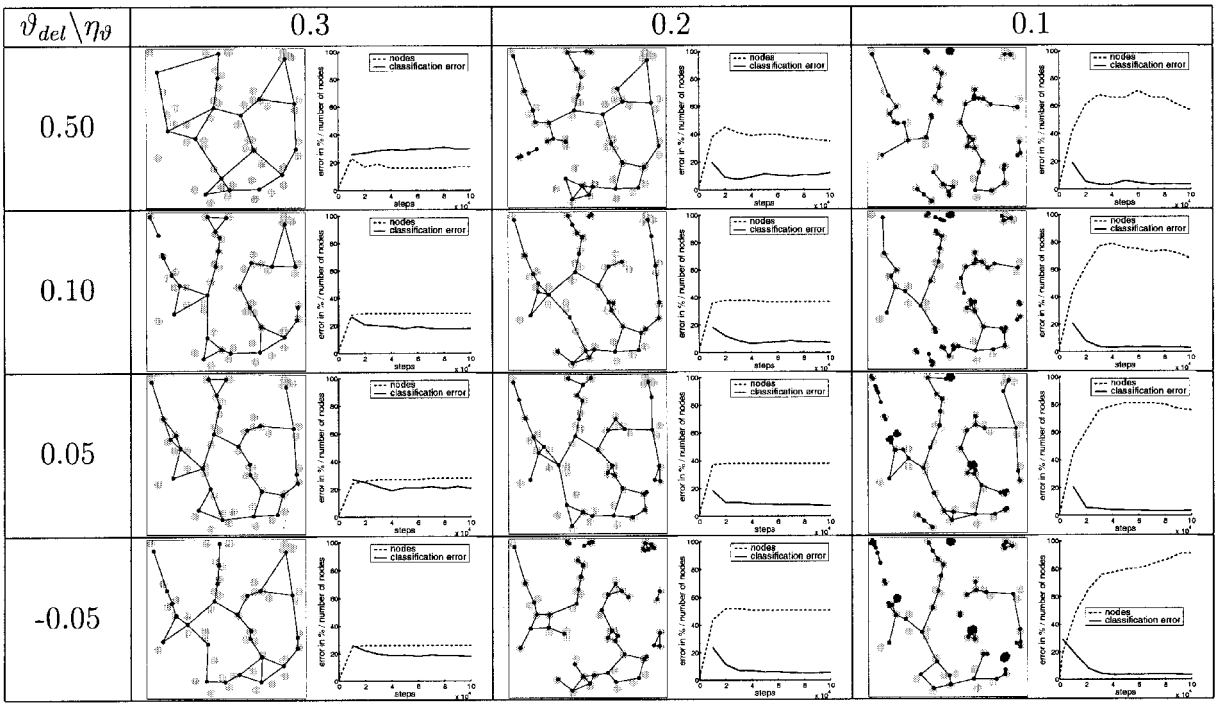
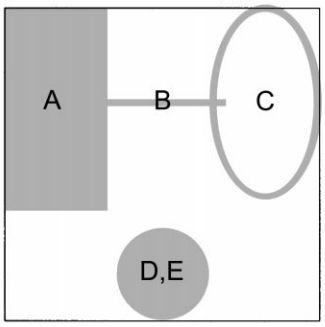


diagram. A newly inserted node always starts on the right side. The initial value of the quality measure for learning B^{L} depends on the nodes which triggered this insertion, since their counters were used for the initialization. In case of a changing environment, the short-term error increases faster than the long-term error, which leads to a higher value of B^{L}, as indicated by course *a* in Fig. 4. After a while the network learns this new situation and B^{L} decreases, but overlappings may prevent an immediate balance of the short- and long-term error. Course *b* illustrates a situation in which a node is inserted between overlapped classes. The short- and long-term error become equal, the quality measure for learning reaches $B^{\text{L}} \approx 1$ and learning is reduced with increasing age (decreasing Y) of the node. The gradient of this cooling process is exponential and depends on the time constant T_Y and on the proportion of the time constants T_{S}, T_{L}. A stationary input probability density always forces the nodes to reach the state $Y \approx 0$; $B^{\text{L}} \approx 1$. Course *c* illustrates a situation of a new node after the environment has just changed and the network quickly solves the problem by learning. In this case, the short-term error can fall below the long-term error. Course *d* shows an old node. A changing environment influences the error and its learning rate is modified in accordance with this change.

3. Illustration with artificial data sets

To illustrate the function of the Life-long Learning Cell Structures, we observe the behavior of the network on a non-stationary input-probability (Fig. 5). An artificial 2D data set is utilized to take advantage of its intuitive manipulation, visualization and its resulting insight into the behavior of the system. For the following tests we employ a paradigm as follows (Hamker & Gross, 1998): from step 1 until 20,000, randomly chosen patterns form four areas (*A*, *B*, *D*, *E*) and three classes are presented with equal probability (environment 1). At step 20,001 the environment changes and patterns form four areas (*A*, *B*, *C*, *E*) and two classes are presented (environment 2). At step 40,001 the environment changes again, etc.

In the first 20,000 steps, the input contains an awful



	Class		Environment (Frequency)					
			1	2	3	4	5	6
A	1	Rectangle	1	1	0	1	1	0
B	1	Line	1	1	1	0	0	1
C	2	Ellipse	0	1	1	1	1	0
D	3	Circular area	1	0	0	0	1	1
E	2	Circular area	1	1	1	0	0	1

Fig. 5. Changing environment composed of five areas (A–E) and three classes. Areas with presented patterns are marked with '1'. The environment changes from 1 to 6 every 20,000 steps. Simulation parameters: $\eta_b = 0.1$, $\eta_n = 0.01$, $\eta_o = 0.15$, $\eta_\vartheta = 0.5$, $T_S = 20$, $T_L = T_Y = T_\vartheta = 100$, $\lambda = 10$, $\vartheta_{age} = 50$, $\vartheta_L^i = 0.05$, $\vartheta_L^o = -0.05$, $\vartheta_{ins} = 0.1$, $\vartheta_{del} = 0.05$, $\vartheta_{delY} = 0.01$, $\vartheta_{delB^L} = 0.01$.

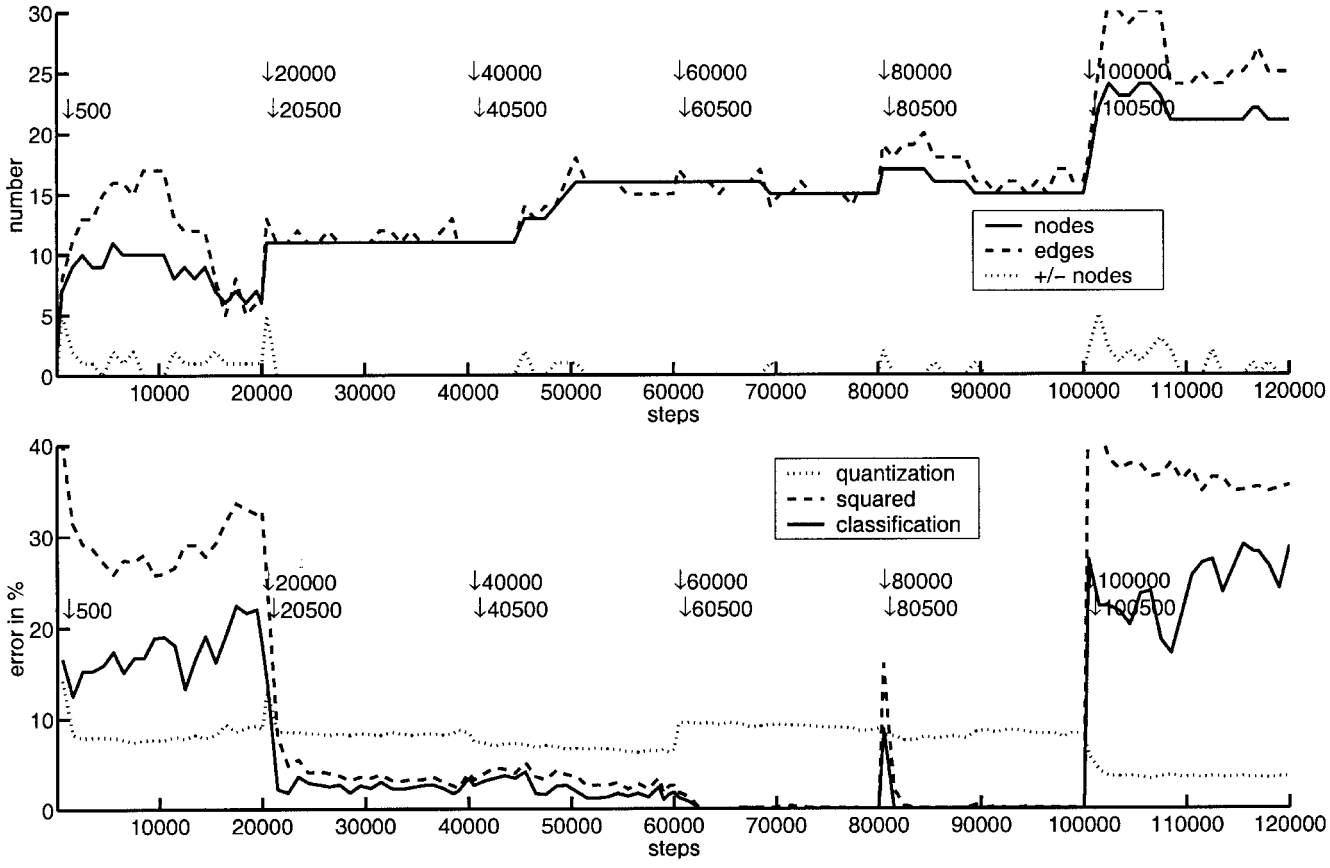


Fig. 6. The number of nodes and the general error measures of the LLCS in environments 1–6. Arrows indicate the number of learning steps at which snapshots from the internal states in the different environments, presented in Figs. 7–9, are taken.

overlap in the circular area which causes a high error. Thus, after 500 steps the internal states of the nodes responsible for the overlapping area show a high long-term error (Fig. 7). As the learning parameter of the input weights expresses, the network is extremely plastic. Nevertheless after 20,000 steps, the algorithm has learned by increasing its insertion threshold, that a further insertion does not improve the squared error and the network stabilizes, as can be seen from the number of nodes (Fig. 6) and the learning parameters (Fig. 7).

Now the environment changes, new errors occur and the algorithm tries to minimize them by changing its weights and inserting new nodes. Although the task gets much easier, there is still an unsolvable overlap between the ellipse and the line that would cause a further insertion of nodes. By increasing the insertion threshold of the relevant nodes, the algorithm learns to stop insertion in the overlapping area. At least after 40,000 steps it has adapted to the environment such that no further learning is needed.

If the probability changes to zero in some regions, such as

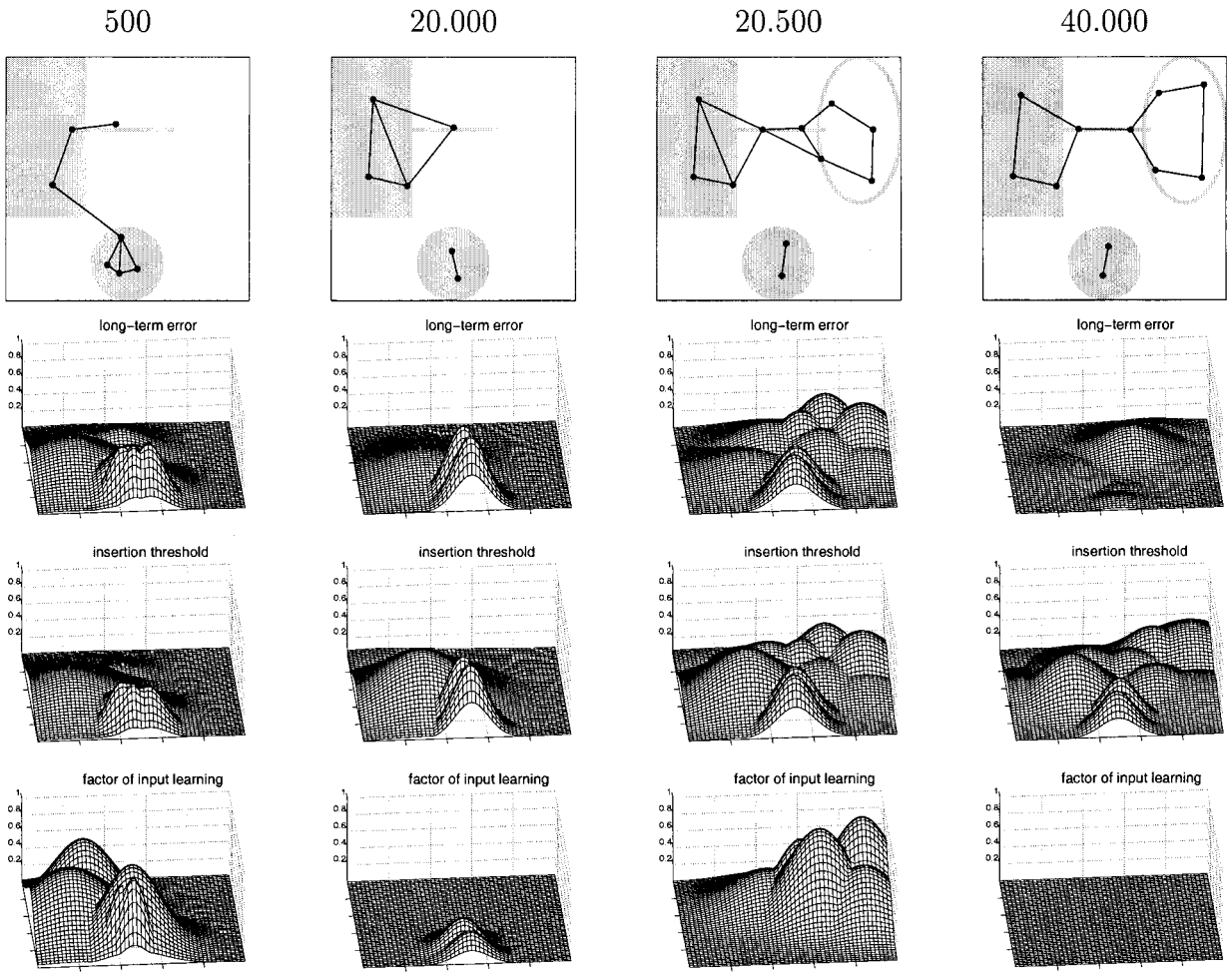


Fig. 7. Internal states of the LLCS in environments 1 and 2. From left to right, two plots show the states in each environment, the first until each 500 learning steps and the second until each 20,000. From the top to the bottom, the input weights w_i, the long-term error τ_L, the insertion threshold τ_ϑ, the factor of the input learning rate α_b^i are presented.

in the environment from 40,000 to 60,000 steps, those remaining nodes, often called 'dead nodes', play a major role in life-long learning (Fig. 8). They are in no way 'dead nodes', instead they preserve the knowledge of previous situations for future decisions. If the old prototype patterns were removed, the knowledge would be lost and the same, already learned situations will again cause errors. Further insertions at the crossing of the line with the ellipse result in a better approximation. However, the number of nodes again becomes stabilized after 50,000 steps (Fig. 6).

In the environment from 60,000 to 80,000 steps, most of the nodes remain at their positions (Fig. 8). The reappearance of area *A* does not raise the error (Fig. 6)—the knowledge was completely preserved. Since the environment shows no overlaps, the error decreases to zero (Fig. 6).

In the environment from 80,000 to 100,000 steps the patterns from the circular area change from class two to class three (Fig. 9). This change of the environment illustrates impressively the localized definition of the stability and plasticity. The network at 80,500 steps remains completely stable aside from nodes covering area *A*. Only here, the network tries to cope with the new situation, inserts new nodes and increases the learning rate. It turns out that the all the new nodes cover the same class and most of them are again deleted. Once more the network stabilizes.

Even serious changes in the environment from 100,000 to 120,000 steps are tolerated. The invertation of the occurrence of patterns in areas *A*, *B*, and *C* does not affect the position of the centers (Fig. 9). The overlap of area *D* and *E* raises the error and the network inserts new nodes, but stabilizes again (Fig. 6).

Summarizing, the algorithm is able to cope with examples of the most important life-long learning scenarios, such as overlaps, never seen inputs, and temporarily not appearing patterns.

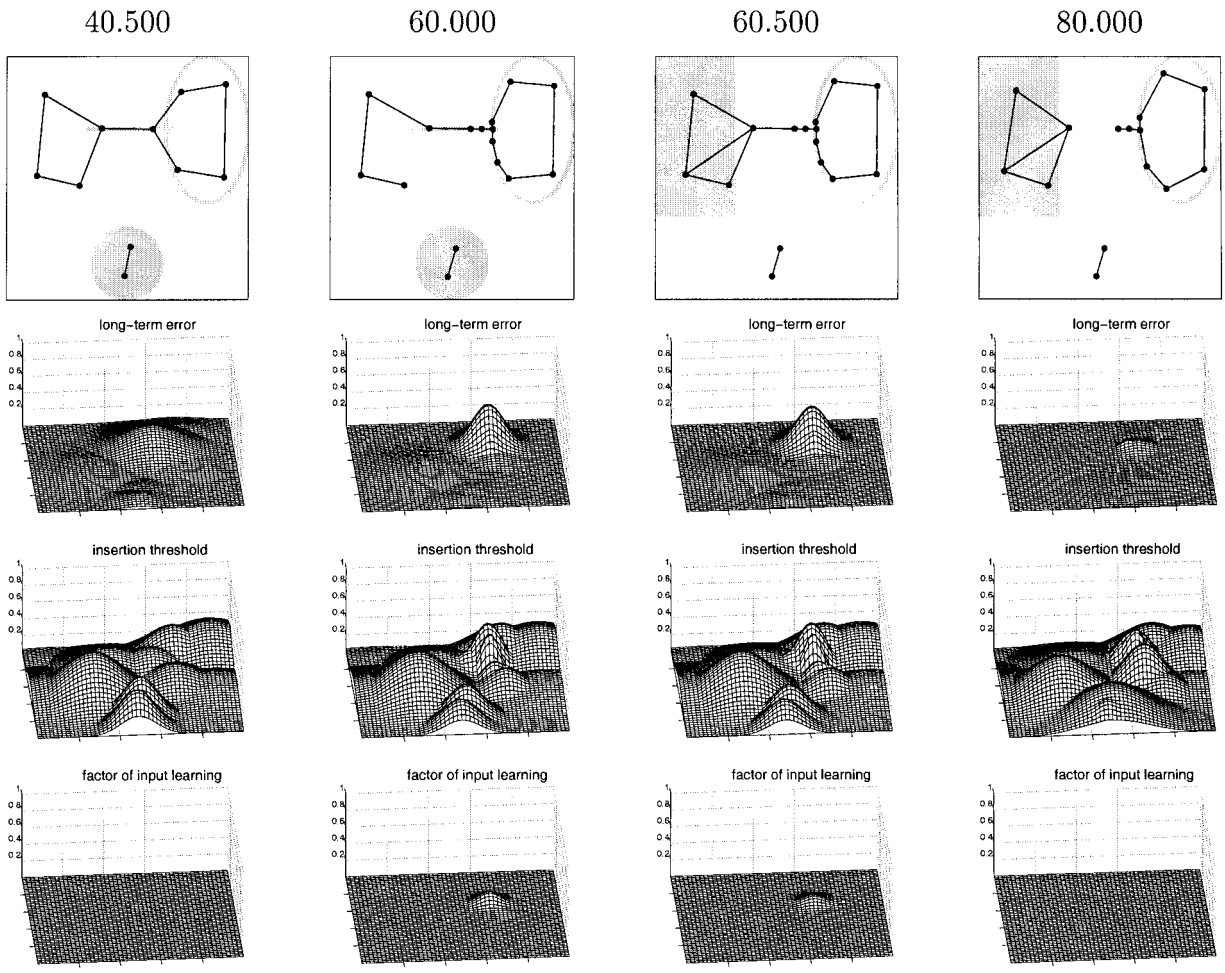


Fig. 8. Internal states of the LLCS in environments 3 and 4. From left to right, two plots show the states in each environment, the first until each 500 learning steps and the second until each 20,000. From the top to the bottom, the input weights w_i, the long-term error τ_L, the insertion threshold τ_ϑ, the factor of the input learning rate α_b^i are presented.

4. Performance evaluation with real data sets

4.1. Stationary distribution

The main purpose of this comparison is to underline that the LLCS automatically stop the insertion of further nodes on real data with a stationary probability distribution and that the performance is as good as those of the GNG. Thus, the behavior of the LLCS on the data sets cancer, diabetes and glass from the PROBEN Benchmark collection (Prechelt, 1994), frequently used for performance comparison (Heinke & Hamker, 1998; Yingwei et al., 1998), is investigated.

4.1.1. Analysis of the data

In the PROBEN Benchmark collection different sets had already been built from three different partitions of the whole data set, which leads to three different mixtures, cancer1, cancer2, and cancer3 (Prechelt, 1994). Each of the mixtures had already been divided into three sets: test set, training set and validation set. The data are analyzed and described in more detail elsewhere (Heinke & Hamker, 1998). Missing values were coded with zero, which is not different from a regular zero. This fact results in several overlappings. cancer is a relatively easy data set which shows some complex boundaries and a few missing values. glass contains only a small number of samples. Thus, the validation and test data do hardly describe the unknown real distribution. Highly overlapped classes and missing values are detected in diabetes.

4.1.2. Performance criteria

The performance of the networks GNG/DCS, GCS, FAM, and MLP is described in Heinke and Hamker (1998). The investigation of the MLP was done by Prechelt (1994) and the results reflect his enormous effort to determine the optimal

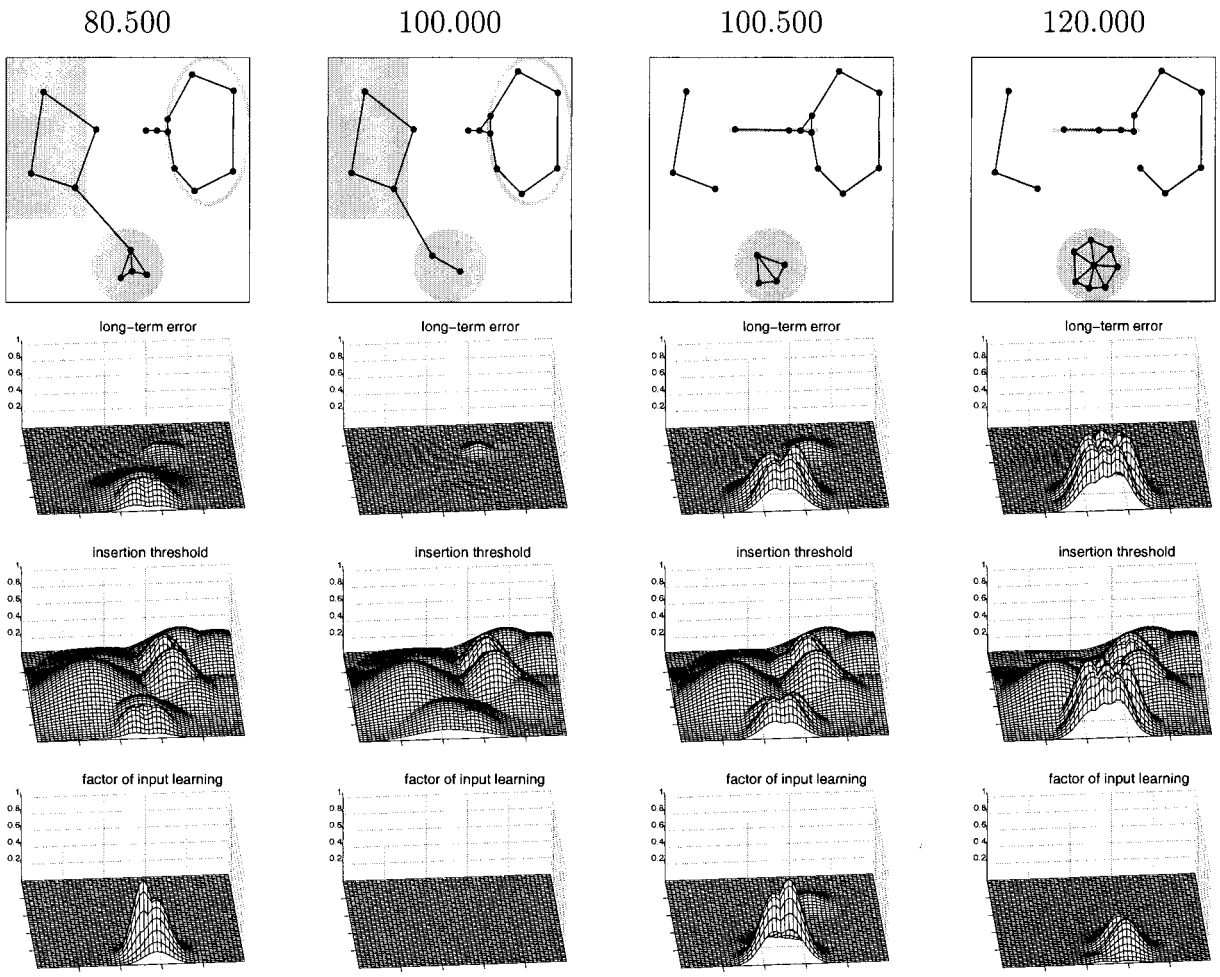


Fig. 9. Internal states of the LLCS in environments 5 and 6. From left to right, two plots show the states in each environment, the first until each 500 learning steps and the second until each 20,000. From the top to the bottom, the input weights w_i, the long-term error τ_L, the insertion threshold τ_ϑ, the factor of the input learning rate α_b^i are presented.

structure of the network. Unlike this comparison, where the stopping of the insertion was determined due to the performance on a validation data set, here the stopping of the insertion is based on the insertion–evaluation cycle without any additional data except from the training data. As an indication for convergence on the data, training was stopped after 15 epochs without any insertion occurring. To investigate the dependency on the insertion and deletion parameters, eight parameter sets were used. For each set, 10 runs with a random initialization of the weights and a different presentation order of the patterns were performed.

4.1.3. Results

In general, the network does not perform much better with parameter sets that lead to more nodes (Fig. 10). Even small networks detect the major characteristics of the distribution. By comparison to other networks, we confirm very good classification results. Although the networks of comparison were selected due to their performance on a validation data set, the LLCS reaches similar results, but without this advantageous selection criterion (Fig. 11). Remarkably, the LLCS achieve these results with much less nodes than used by GNG and GCS and similar or even fewer than used by FAM as reported by Heinke and Hamker (1998).

Without any specified assumption, the LLCS insert an appropriate number of nodes. They self-stabilize on a stationary distribution and reach a performance equal to networks such as GNG, GCS, FAM and MLP selected by cross-validation. Further figures, such as the course of the error and the number of nodes plotted over the learning steps, are reported in Hamker (1999).

4.2. Non-stationary distribution

Relevant issues for the application of the LLCS deal with

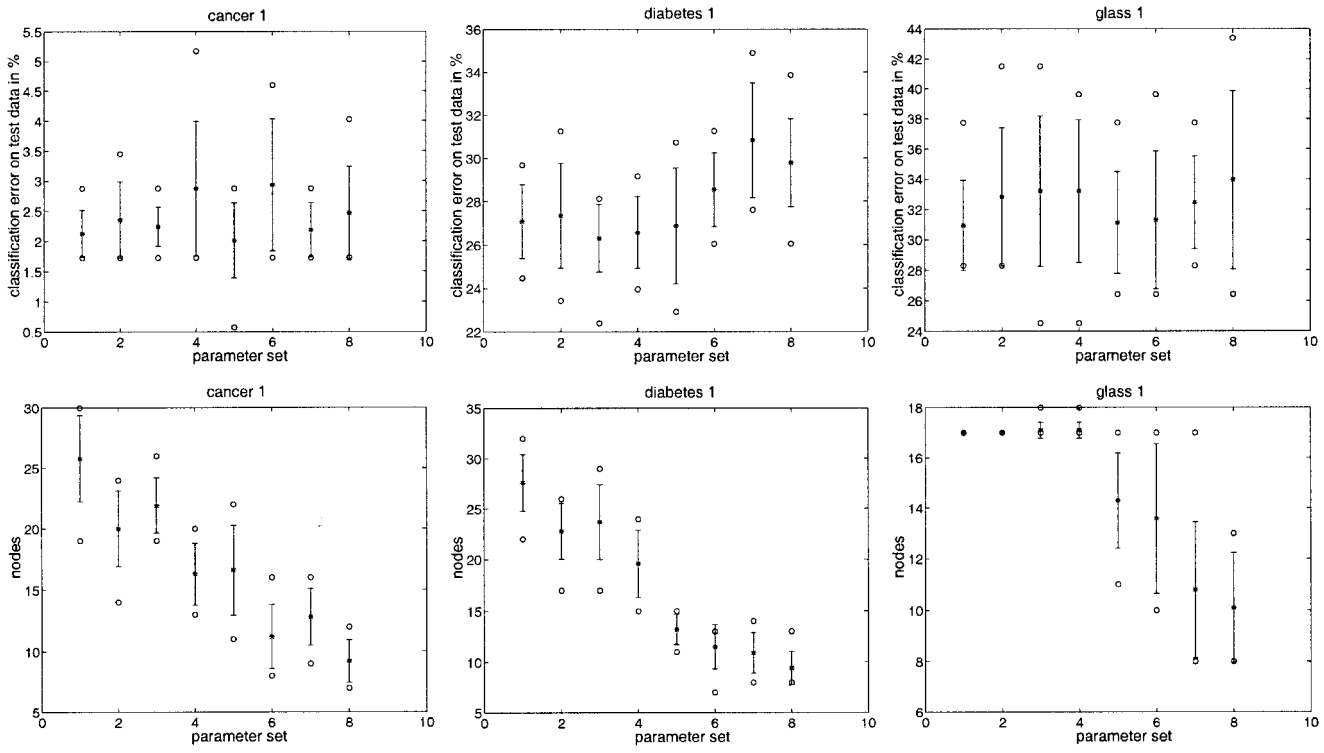


Fig. 10. Mean, stdv., best, worst error and the number of nodes of the LLCS for eight examined parameter sets on the test data (only the first of each data set is shown). As discussed in Section 2.3 more conservative parameter settings like parameter set 5–8 only detect the rudimentary class distribution. Nevertheless, this is sufficient. Even networks with very few nodes achieve quite good results. This again underlines the parameter insensitivity and confirms the algorithm's strategy to firstly learn the general distribution and then detect the details. Parameter set: 1: = $\{\eta_\vartheta = 0.1,\ \vartheta_{\text{ins}} = 0.3,\ \vartheta_{\text{del}} = 0.2\}$; 2: = $\{\eta_\vartheta = 0.1,\ \vartheta_{\text{ins}} = 0.3,\ \vartheta_{\text{del}} = 0.4\}$; 3: = $\{\eta_\vartheta = 0.1,\ \vartheta_{\text{ins}} = 0.4,\ \vartheta_{\text{del}} = 0.2\}$; 4: = $\{\eta_\vartheta = 0.1,\ \vartheta_{\text{ins}} = 0.4,\ \vartheta_{\text{del}} = 0.4\}$; 5: = $\{\eta_\vartheta = 0.2,\ \vartheta_{\text{ins}} = 0.3,\ \vartheta_{\text{del}} = 0.2\}$; 6: = $\{\eta_\vartheta = 0.2,\ \vartheta_{\text{ins}} = 0.3,\ \vartheta_{\text{del}} = 0.4\}$; 7: = $\{\eta_\vartheta = 0.2,\ \vartheta_{\text{ins}} = 0.4,\ \vartheta_{\text{del}} = 0.2\}$; 8: = $\{\eta_\vartheta = 0.2,\ \vartheta_{\text{ins}} = 0.4,\ \vartheta_{\text{del}} = 0.4\}$. Fixed parameters: $\eta_b = 0.1$, $\eta_n = 0.01$, $\eta_o = 0.15$, $T_{\text{S}} = 20$, $T_{\text{L}} = T_Y = T_\vartheta = 100$, $\lambda = 10$, $\vartheta_{\text{age}} = 50$, $\vartheta_{\text{L}}^{\text{i}} = 0.05$, $\vartheta_{\text{L}}^{\text{o}} = -0.05$, $\vartheta_{\text{del}Y} = 0.01$, $\vartheta_{\text{del}B^{\text{L}}} = 0.01$.

non-stationary input probability distributions. Thus, a benchmark based on real data was designed, which contains major demands on life-long learning. It consists of 10 data sets with 29 features and four classes. Each data set was built from four images which contain four different materials (classes), such as journals, cardboard, newspaper and others, recorded under different lightning conditions. Each pattern is gained from a color histogram of a tile sized 32×32 pixels. For instance we have an image of 512×512 pixels which contains journals. We take each tile and measure for each pixel the distance to 29 color prototypes, that were distributed in the three dimensional color space. The feature of a tile is a vector of the average distance of all pixels to the color prototypes. Thus, for each tile we compute a feature vector that represents the covered material. For details about the data and the feature extraction see Hamker, Debes, Pomierski and Gross (1998).

One after another, a data set is presented just once within 402 blocks each of 10 patterns belonging to the same class so that the environment continuously changes. For example, we have a data set gained from four images (two with journals, one with cardboard, and one with newspaper). We randomly choose an image and select 10 tiles. The selected tiles were disregarded for further presentations. As an example of a changing environment, this process simulates the typical process of material on a conveyor belt, where a vision system selects grouped regions of similar features.

The output of the network on all data sets is recorded in parallel (Fig. 12). This means the impact of training a particular data set on the performance of another data set can be analyzed. According to the correspondence between different data sets, learning in one environment is of advantage to some environments, whereas others suffer from strong overlaps. This procedure is repeated using four different parameter sets.

In this experimental context, a neural network shows plasticity if it can learn a new data set as good as a neural network that was not confronted with different data sets before. A neural network shows stability if it is able to preserve the old prototype patterns from a previous data set, but only if they are not contradictory—i.e. relearning of prototypes should only occur if they are inconsistent with the current data.

4.2.1. Analysis of the data

Each data set contains 4096 samples. The overlap of different data sets can be estimated by a confusion matrix, which indicates to what extent feature vectors are assigned to a wrong class. We extended this concept to estimate the confusion of a trained data set to a non-trained data set. Thus, the LLCS were

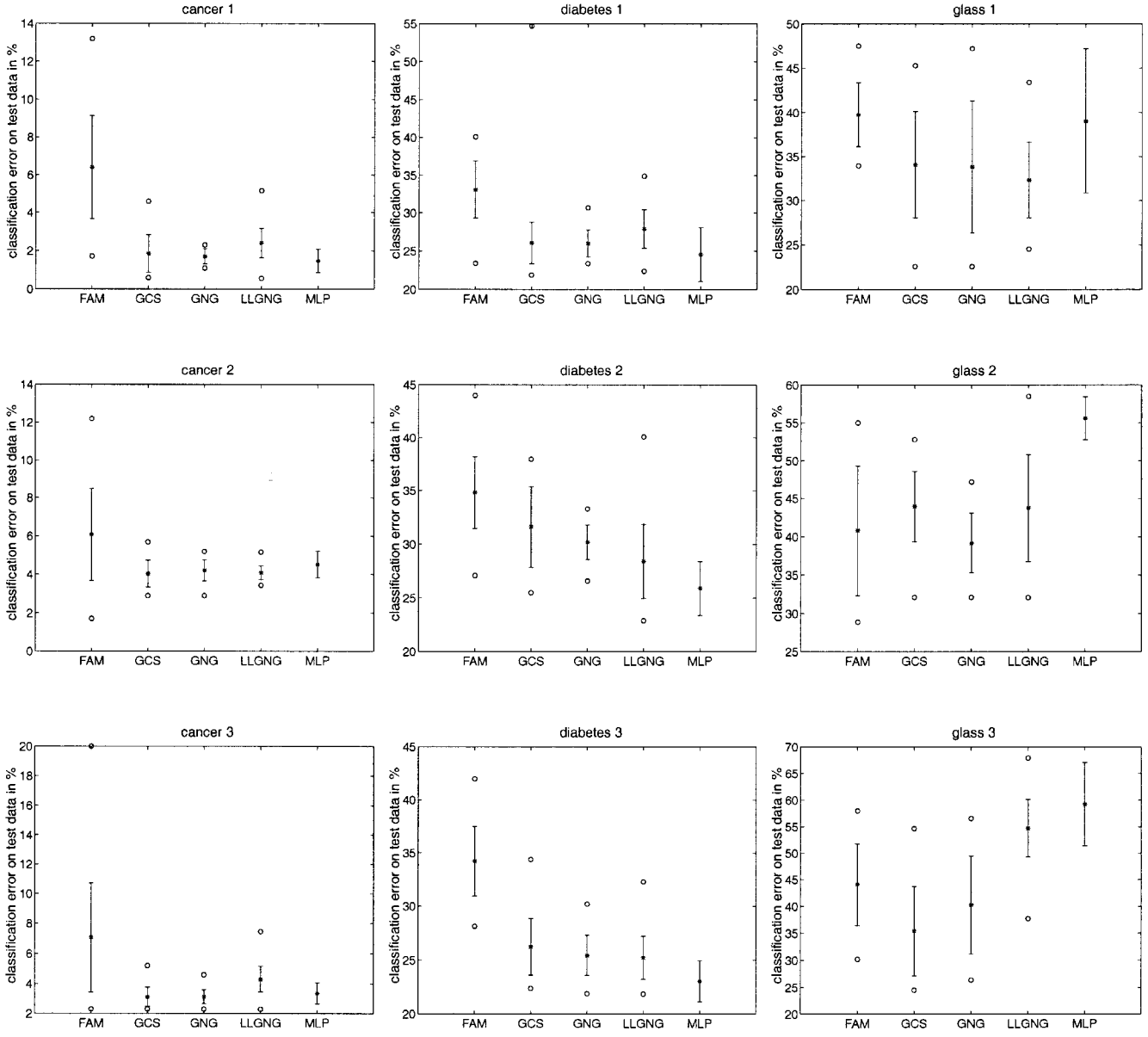


Fig. 11. Mean test error, stdv., best and worst net of the LLCS in comparison to the networks analyzed in a previous benchmark (Heinke & Hamker, 1998) (for MLP, Prechelt, 1994, best and worst are not plotted).

trained for three epochs on a particular data set and the confusion matrix of all non-trained data sets is estimated. In order to achieve a more precise criterion, all patterns that result in a maximal output activation lower than a threshold $\gamma \in \{0.1, 0.3, 0.5\}$ were declared as a not sufficiently learned pattern with a separate class (new), instead of being assigned to the class with the largest output activation.

The data show a strong overlap between the classes (Fig. 13) and abrupt as well as slow changes in the temporal presentation. From the first data set to the third only slow changes emerge, whereas afterwards abrupt changes occur.

As we can see from Fig. 13, several data sets do not contain all classes. Especially, data set four contains only patterns of one class that has a strong overlap with different classes in other data sets. This is the reason for the large increase of error when data set four is selected as training data (Fig. 12). Particularly, the error on data set six increases strongly, because both data sets share no common class. Only the performance on data set nine can profit by the training on data set four, because both share similar patterns in each class (Fig. 13).

4.2.2. Analysis of plasticity

Plasticity defines that a neural network is capable to learn new patterns. The general performance to learn a data set was shown in Section 4.1. Thus, this analysis has to demonstrate the network's capability to learn a new data set after different data sets were presented. For a quantitative

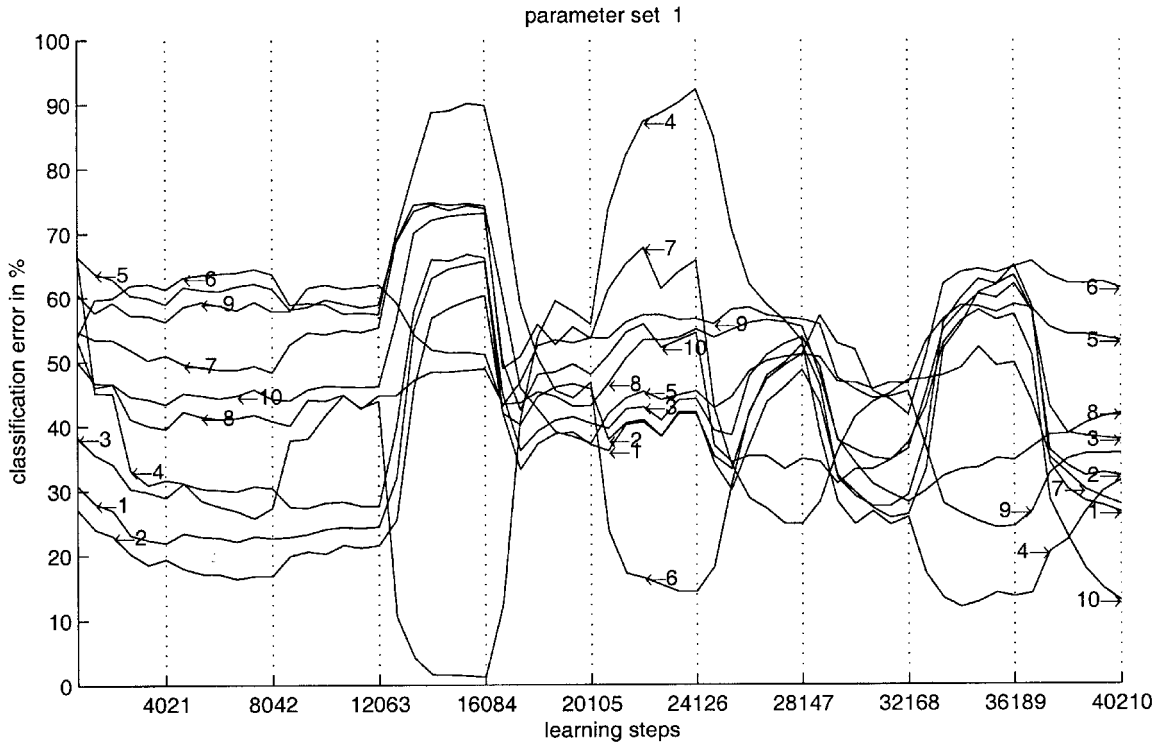


Fig. 12. Average classification error (parameter set 1) of data sets 1–10 gained from 10 different runs with non-stationary data. Parameters: 1: = $\{\eta_\vartheta = 0.1, \vartheta_{\text{del}} = 0.2\}$; 2: = $\{\eta_\vartheta = 0.1, \vartheta_{\text{del}} = 0.4\}$; 3: = $\{\eta_\vartheta = 0.2, \vartheta_{\text{del}} = 0.2\}$; 4: = $\{\eta_\vartheta = 0.2, \vartheta_{\text{del}} = 0.4\}$. Fixed parameters: $\eta_b = 0.8$, $\eta_n = 0.01$, $\eta_o = 0.01$, $T_{\text{S}} = 100$, $T_{\text{L}} = T_Y = T_\vartheta = 200$, $\lambda = 100$, $\vartheta_{\text{age}} = 60$, $\vartheta_{\text{ins}} = 0.4$, $\vartheta_{\text{L}}^{\text{i}} = 0.1$, $\vartheta_{\text{L}}^{\text{o}} = -0.05$, $\vartheta_{\text{del}Y} = 0.01$, $\vartheta_{\text{del}B^{\text{L}}} = 0.01$.

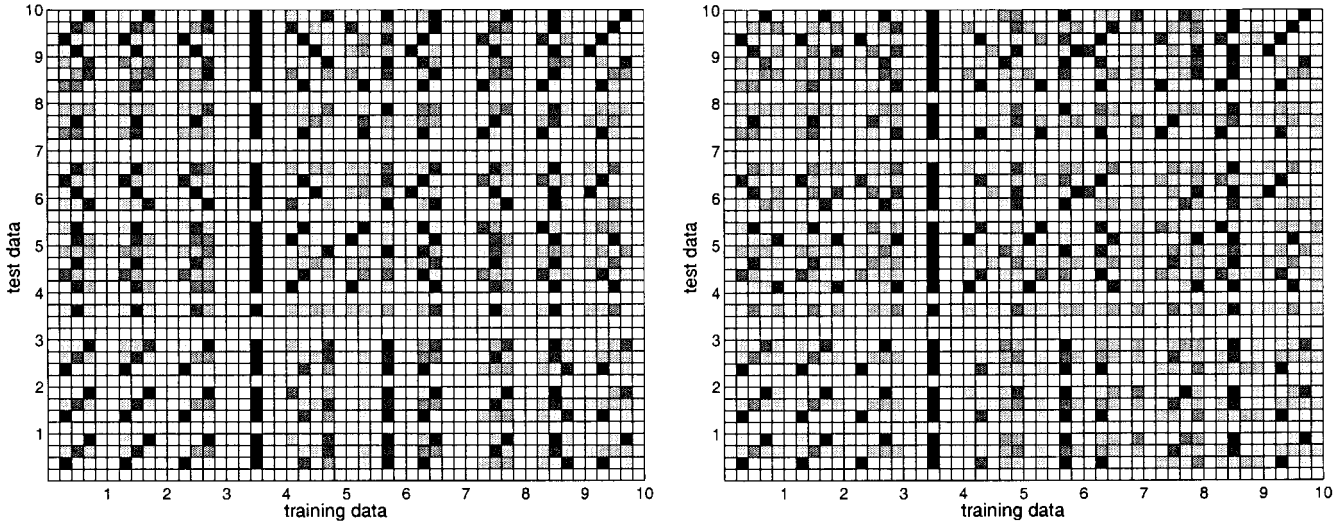


Fig. 13. Confusion matrix of all data sets with $\gamma = 0.1$ (left) and $\gamma = 0.5$ (right). See Fig. 16 for the structure of the matrix. A black rectangle indicates a large value. Data sets with no overlap (high similarity) show a black diagonal within each 4×5 array. Activations at other areas indicate a complete overlap. The larger threshold γ results in a bigger class C_{new}.

evaluation, two bounds were defined. The upper bound was gained by training the network just once on a data set. The lower bound was achieved by training the network on the data set as long as the tested network was trained on different data sets. In the latter condition, the number of nodes is similar. Each bound is an average from 10 runs always using the first parameter set.

The error cannot fall below the lower bound, though the case is different with the upper bound. Here, the training process of a just initialized network is compared to a network that was learned on different data sets before. Thus, achieving the upper bound is a good performance, but an error lower than the upper bound is even better.

The course of errors in Fig. 14 demonstrates a good

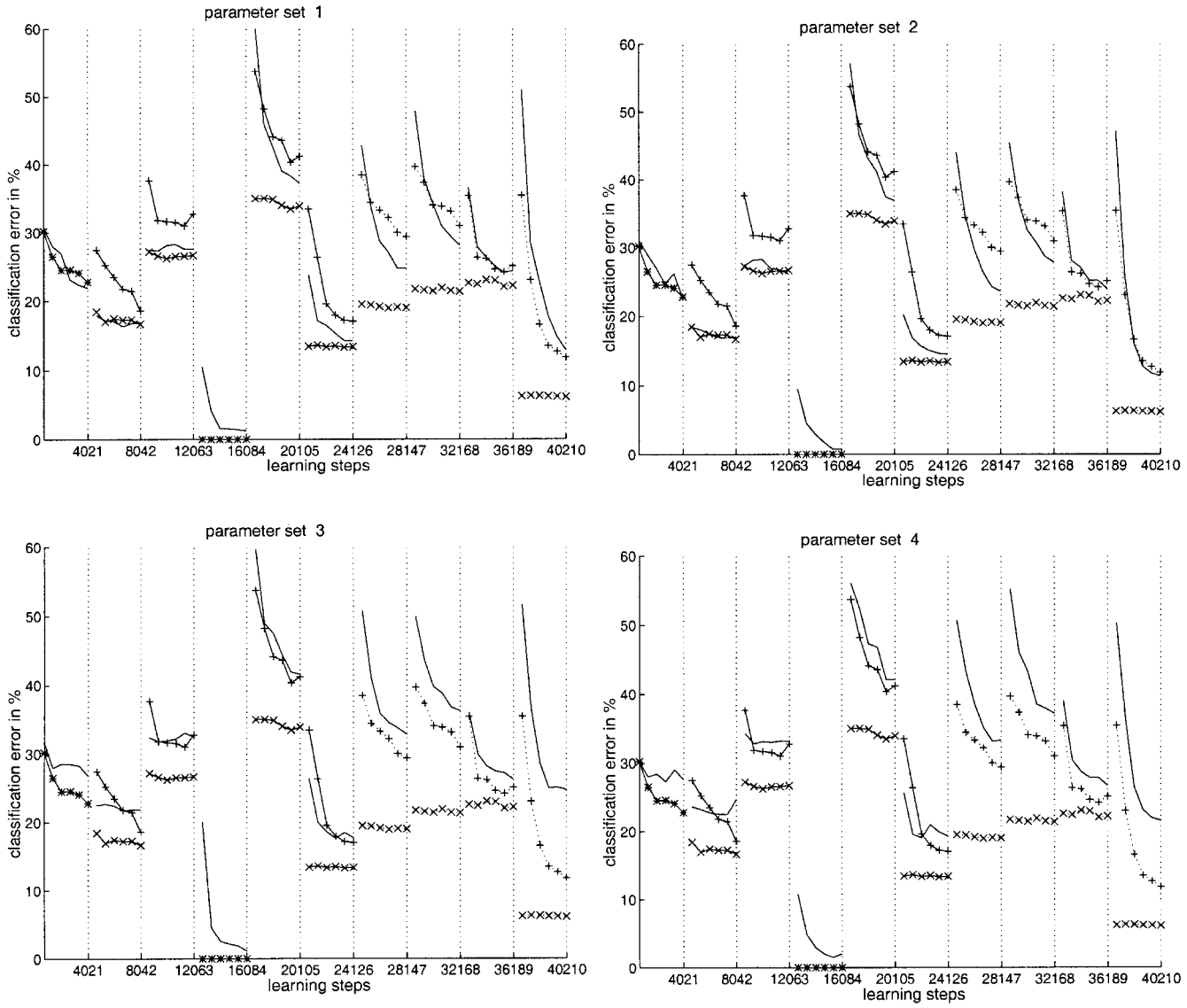


Fig. 14. Results of the plasticity analysis. The error of the network is shown when presented a never seen data set (—), compared to the upper bound (+) and the lower bound (×).

plasticity performance. After the environment changes, the error decreases, and in most cases falls below the upper bound. Even more conservative parameter sets (sets 3 and 4) show a large decrease in the error, although these networks operate with less than half of the nodes (Fig. 15).

4.2.3. Analysis of stability

Stability is the crucial aspect of the Stability–Plasticity Dilemma. The illustration on artificial data has demonstrated the capability to preserve the prototype pattern. Nevertheless, if the environment changes, the new situation can be inconsistent with the current situation, and the network has to adapt. This means it loses some previous knowledge in order to respond accurately.

Thus, an analysis of stability on real data cannot rely on simply measuring the amount of forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990). Likewise, the amount of time to relearn the pattern (Heterhington & Seidenberg, 1989) is not the best criterion, because this depends too much on the algorithm's learning strategy. An appropriate measure has to consider the correspondence and the overlap of successive environments. In addition to the data analysis, we have to find a measure that includes the overlap of following environments and indicates which data should be preserved. To accomplish this, the categorization into 'new' patterns is helpful. In this line, a neural network shows stability after a change in the environment if the classification error is smaller than the bound that results from the correspondence of the data of following environments plus the patterns that were assigned to class C_{new} (Fig. 16):

$$\overline{S_\gamma} = 100\% - (\#\text{correspondence} + \#\text{new}_\gamma)\% \tag{25}$$

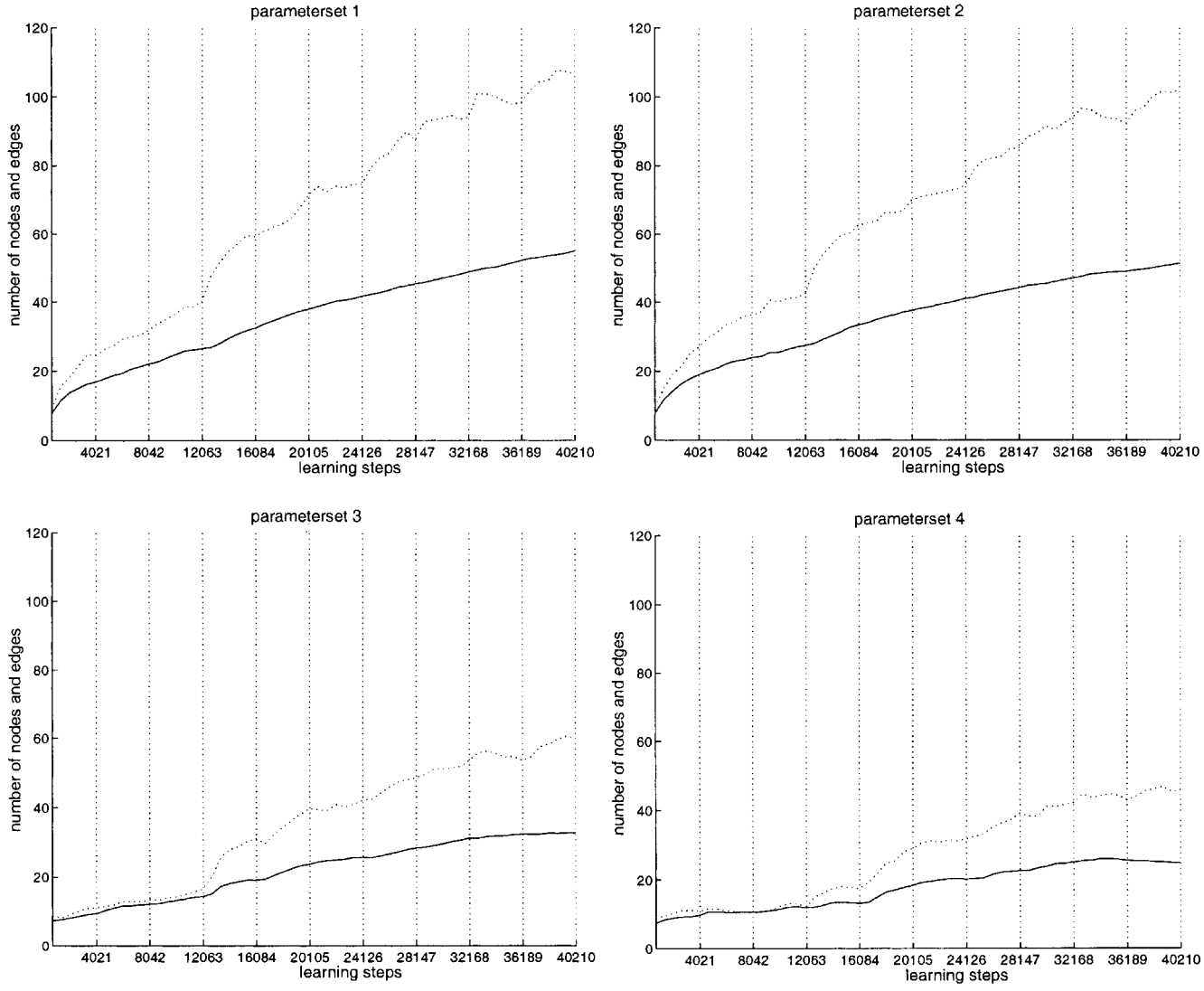


Fig. 15. The number of nodes (—) and edges (···) of the network when presented the 10 different data sets one after another.

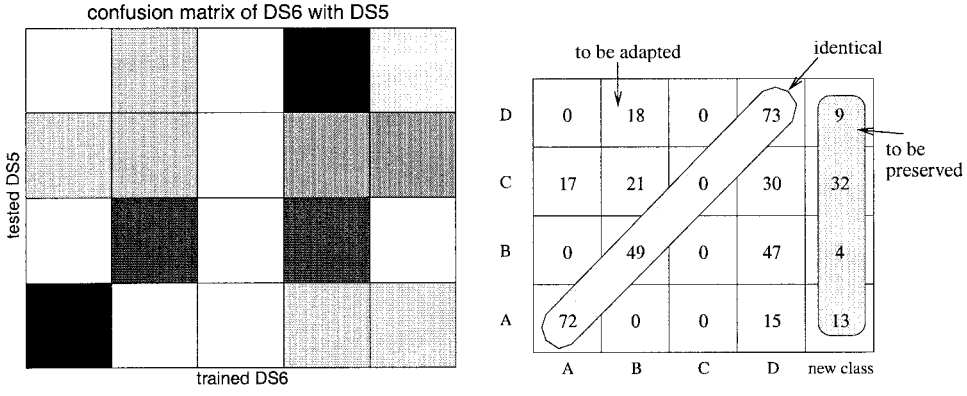


Fig. 16. Left: extract from the confusion matrix (Fig. 13) with $\gamma = 0.5$. The cut shows training data set 6 and test data set 5. Right: evaluation of the confusion matrix. The values on the main diagonal mark the correspondence of the data sets. Even if the data set 5 was not presented as a training set, the patterns of the main diagonal could be classified correctly. The patterns of other fields except for the class C_{new} could probably not be preserved from relearning, because these patterns from data set 6 collide with data set 5, but the amount of pattern from class C_{new} could be expected to be preserved when the network is trained on data set 6, because, although there might be an overlap, the patterns are less similar. For the above example the stability boundary $\overline{S_{0.5}}$ results in $\overline{S_\gamma} = 100\% - (194\% + 58\%)/4 = 37\%$.

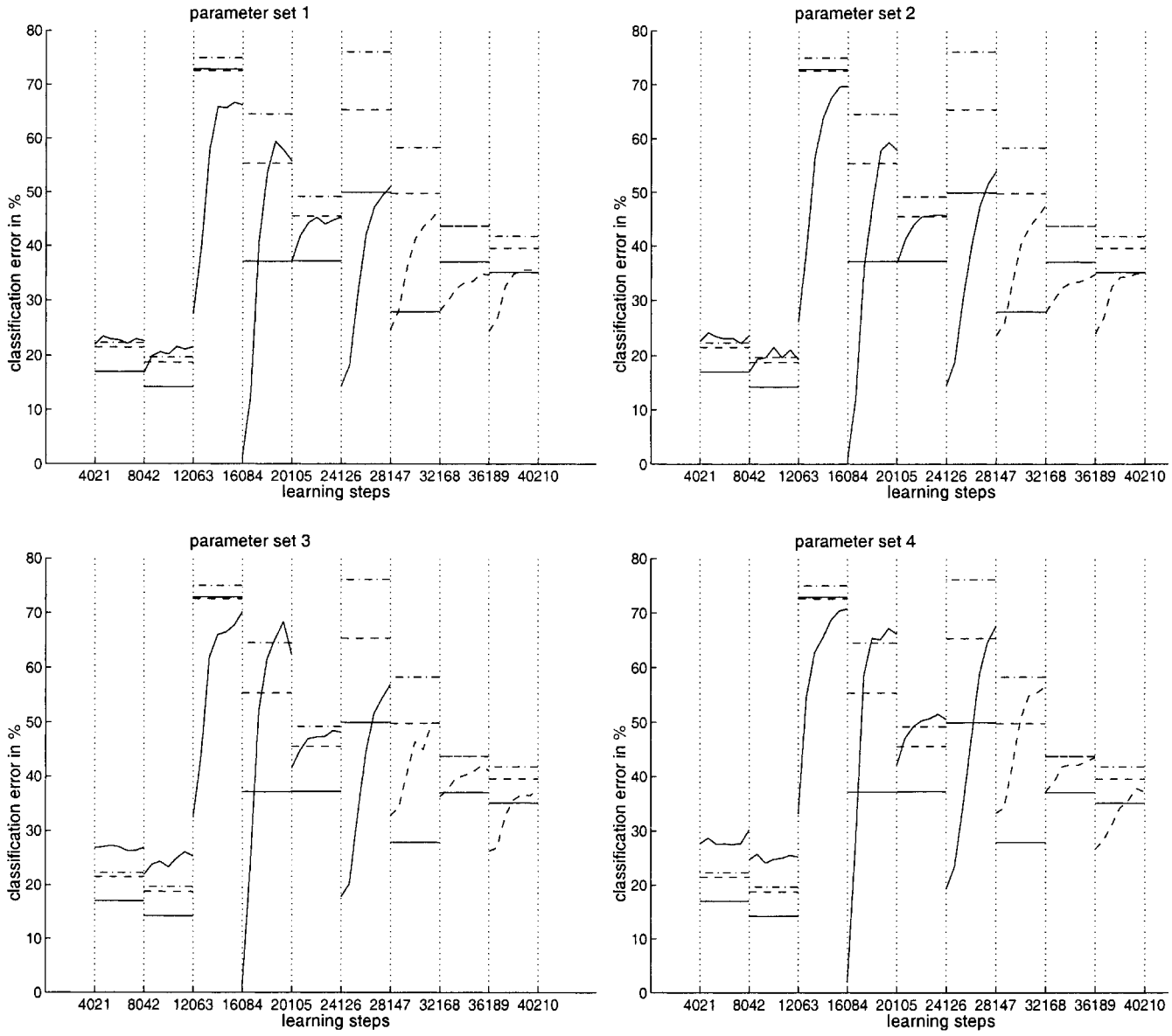


Fig. 17. Results of the stability analysis. The figure shows the stability bounds estimated with $\gamma = 0.1$ (···), $\gamma = 0.3$ (– –), and $\gamma = 0.5$ (—), based on a network trained with parameter set 1 compared to the error on the previously trained data (—), e.g. set 7 is training data and set 6 is observed. On the first two data sets the error is above the estimated bounds. This is due to the low number of nodes created so far. In many cases within parameter set 1 the error is below the lowest bound, which underlines the exceptional stability property of the LLCS, but even the networks trained with the more conservative and less node producing parameter sets show a good stability performance. Only in some cases, mostly if the training on the previous data set does not decrease the error to the level of the reference network, which determines the bound, the error exceeds some bounds, but even then, the networks still preserve some knowledge.

For different values of the variable γ, different bounds emerge. The more the classification error remains lower than these bounds, the more stable is the network when the environment changes (Fig. 17).

After the change from data set 1 to data set 2 the classification error is larger than the bound, but this is merely based on the short training time and the low number of nodes, which hinder a precise representation of the data. The further course clearly indicates the preservation of old prototype patterns.

5. Conclusions

Continuous learning must solve the catastrophic interference, which mainly occurs in a distributed representation, but also localist or partly distributed representations suffer from interference, especially if the network has not sufficient nodes. Thus, we discussed different strategies to insert new nodes in incremental neural networks. A similarity based insertion as used in ART networks (Carpenter & Grossberg, 1987) and also in RBF networks (Berthold &

Diamond, 1995; Platt, 1991; Yingwei et al., 1998) has the advantage of increasing the number of nodes without an instability concerning insertion. However, this advantage is at the expense of a low performance and a high number of nodes, because the optimal similarity is unknown, not equally distributed within the input space, and may change over time. If an error signal exists, it would be better to use it for insertion. While a global error based insertion (Fahlman & Lebiere, 1990) is useless for life-long learning, because the error changes over time, a promising approach is a local error based insertion criterion as used by Fritzke (1992) or also in ARTMAP (Carpenter et al., 1991) triggered by the Inter-ART-Reset. But this raises the question, how to suppress insertion in overlapping decision areas, where errors always occur. Furthermore, life-long learning has to address the question in which cases the weights of the network have to be adapted to learn new patterns and when the weights should not change to guarantee stability.

Our evolved neural network is based on previous work by Fritzke (1992, 1995); Bruske and Sommer (1995); Bruske (1998) and extends the ability of the Cell Structures to locally adapt its stability and plasticity—for learning and for insertion. The essential innovation compared to that previous work is the self-adjustable balance between the extremes of the Stability–Plasticity Dilemma by adapting the learning rate and the insertion capability of each node separately. This allows the network to self-stabilize for a stationary probability density of the input patterns and to switch locally to plasticity if relevant changes occur—a framework useful as a general strategy of growing RBF networks.

The empirical investigations for the example of supervised learning with real data confirm no decrease in classification performance as compared to the Cell Structures. The choice of parameters allows a gradual change towards the original behavior of the Cell Structures. Thus, the Life-long Learning Cell Structures extend the quality of the Cell Structures regarding a favorable compromise to the Stability–Plasticity Dilemma, which is characterized as:

- The stability and plasticity is defined locally in the network, i.e. for each center.
- The stability and plasticity concerns the adaptation of the centers, the learning of decision boundaries and the number of centers.
- The number of nodes is not predefined—instead an adequate number is learned by continuously adapting local insertion thresholds according to the performance of the network on the data.
- In the case of a stable state, local plasticity only occurs due to relevant changes in the input probability density, i.e. changes in the error probability density.

Of course, the need for continuously supervised learning systems is rare. To give an example, we applied the network in a visual sorting prototype for wastepaper (Hamker et al., 1998). The idea was to select valuable material from the rest by gripping the material with a manipulator. To avoid time-consuming wrong selections, especially if environmental or material changes occur, the visual system, i.e. the network, was continuously trained by a largely reliable tactile sensor. Irrespective of this specific engineering task, our aim here is to foster the concept of life-long or continuous learning. Major applications can be spotted in the time series prognosis and robotics domains. Thus, we would like to stress the fact that without fundamental changes, simply by using another learning scheme for the output layer, the algorithm can be adapted for autonomous process control by using local linear maps similar to Martinetz, Berkovich, and Schulten (1993) or for autonomous robots by incorporating reinforcement learning similar to Bruske et al. (1998) and Gross, Stephan and Boehme (1996).

Although much still has to be done, incremental networks embedded within a performance estimation to control the number of nodes and the learning parameters offer a serious approach for systems that act in changing environments.

Acknowledgements

The body of this work was performed at the Department of Neuroinformatik, Technische Universität Ilmenau (Germany). I thank Prof. H.-M. Gross for his support and T. Vesper for his fruitful discussions and his implementation of an initial version of the algorithm in his diploma thesis. I would also like to thank D. Surmeli and the reviewers for their helpful comments.

References

Ahrns, I., Bruske, J. & Sommer, G. (1995). On-line learning with Dynamic Cell Structures. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 141–146).

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, *16*, 299–307.

Berthold, M. R. & Diamond, J. (1995). Boosting the performance of RBF networks with dynamic decay adjustment. In *Advances in neural information processing systems (NIPS 7)* (pp. 521–528). Cambridge: MIT Press

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Bruske, J., & Sommer, G. (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, *7*, 845–865.

Bruske, J., Hansen, M., Riehn, L., & Sommer, G. (1996). Adaptive saccade control of a binocular head with Dynamic Cell Structures. *In Proceedings of the International Conference on Artificial Neural Networks (ICAN'96)*, 215–220.

Bruske, J. (1998). *Dynamische Zellstrukturen. Theorie und Anwendung eines KNN-Modells*. PhD Thesis, Technische Fakultät der Christian Albrechts-Universität zu Kiel.

Bruske, J., Ahrns, L., & Sommer, G. (1998). An integrated architecture for learning of reactive behaviors based on dynamic cell structures. *Robotics and Autonomous Systems*, *22*, 87–102.

Carpenter, G. A., & Grossberg, S. (1987). ART2: Self-organisation of

stable category recognition codes for analog input patterns. *Applied Optics*, *26*, 4919–4930.

Carpenter, G. A., Grossberg, S., & Reynolds, J. H. (1991). ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, *4*, 543–564.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, *3*, 698–713.

Chen, Y. Q., Thomas, D. W., & Nixon, M. S. (1994). Generating–shrinking algorithm for learning arbitrary classification. *Neural Networks*, *7*, 1477–1489.

Eriksson, P. S., Perfilieva, E., Bjork-Eriksson, T., Alborn, A. M., Nordborg, C., Peterson, D. A., & Gage, F. H. (1998). Neurogenesis in the adult human hippocampus. *Nature Medicine*, *4*, 1313–1317.

Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in neural information processing systems 2* (pp. 524–532). Los Altos, CA: Morgan Kaufmann Publishers.

Freeman, J. A. S., & Saad, D. (1997). On-line learning in radial basis function networks. *Neural Computation*, *9*, 1601–1622.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, *3*, 128–135.

Fritzke, B. (1992). *Wachsende Zellstrukturen—ein selbstorganisierendes neuronales Netzwerkmodell*. PhD Thesis, Technische Fakultät der Universität Erlangen-Nürnberg.

Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*, 1441–1460.

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in neural information processing systems (NIPS 7)* (pp. 625–632). Cambridge, MA: MIT Press.

Fritzke, B. (1997). A self-organizing network that can follow non-stationary distributions. In *Proceedings of the International Conference on Artificial Neural Networks (ICAN'97)* (pp. 613–618). Springer.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1–58.

Gross, H.-M., Stephan, V. & Boehme, H.-J. (1996). Sensory-based robot navigation using self-organizing networks and Q-learning. In *Proceedings of the World Congress on Neural Networks (WCNN'96), San Diego* (pp. 94–99).

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, *23*, 121–134.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks*, *1*, 17–61.

Grossberg, S., & Merrill, J. W. L. (1996). The hippocampus and cerebellum in adaptively timed learning, recognition and movement. *Journal of Cognitive Neuroscience*, *8*, 257–277.

Hamker, F. H. & Gross, H.-M. (1997). Task-based representation in lifelong learning incremental neural networks. In *VDI Fortschrittberichte, Reihe 8, Nr. 663, Workshop SOAVE'97, Ilmenau* (pp. 99–108).

Hamker, F. H. & Gross, H.-M. (1998). A lifelong learning approach for incremental neural networks. In *Proceedings of the Fourteenth European Meeting on Cybernetics and Systems Research (EMCSR'98), Vienna* (pp. 599–604).

Hamker, F., Debes, K., Pomierski, T., Gross, H.-M., (1998). *Multisensorielles Integriertes Realzeit Inspektions-System MIRIS: Lösung der MIKADO-Sortieraufgabe. Schriftenreihe des FG Neuroinformatik der TU Ilmenau*, Report. 2/98.

Hamker, F. H. (1999). *Visuelle Aufmerksamkeit und lebenslanges Lernen im Wahrnehmungs-Handlungs-Zyklus*. PhD Thesis, Technische Universität Ilmenau.

Hamker, F.H., Paetz, J., Thöne, S., Brause, R., Hanisch, E., (2000). *Erkennung kritischer Zustände von Patienten mit der Diagnose 'Septischer Schock' mit einem RBF-Netz*. Report. 4/2000, Franfurt am Main: Institute of Imformatik, JW Goethe Iniversität.

Heinke, D., & Hamker, F. H. (1998). Comparing neural networks: a benchmark on growing neural gas, growing cell structures, and fuzzy ARTMAP. *IEEE Transactions on Neural Networks*, *9*, 1279–1291.

Heskes, T. M. & Kappen, B. (1993). On-line learning processes in artificial neural networks. In *Mathematical foundations of neural networks* (pp. 199–233). Amsterdam: Elsevier Science Publishers.

Hetherington, P. & Seidenberg, M. (1989). Is there 'catastrophic interference' in connectionist networks? In *Proceedings of the 11th Annual Conference of the Cognitive Science Society* (pp. 26–33). Hillsdale, NJ: LEA.

Karayiannis, N. B., & Mi, G. W. (1997). Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, *8*, 1492–1506.

Karayiannis, N. B. (1999). Reformulated radial basis neural networks trained by gradient descent. *IEEE Transactions on Neural Networks*, *10*, 657–671.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.

Lim, C. P., & Harrison, R. F. (1997). An incremental adaptive network for on-line supervised learning and probability estimation. *Neural Networks*, *10*, 925–939.

Martinetz, T. M. & Schulten, K. J. (1991). A 'neural gas' network learns topologies. In *Artificial neural networks* (vol. I, pp. 397–402). Amsterdam: North Holland.

Martinetz, T. M., & Schulten, K. J. (1994). Topology representing networks. *Neural Networks*, *4*, 507–522.

Martinetz, T. M., Berkovich, S., & Schulten, K. J. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*, 558–569.

McClelland, J., McNaughton, B., & O'Reilly, R. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the success and failures of connectionist models of learning and memory. *Psychological Review*, *102*, 419–457.

McCloskey, M., & Cohen, N. (1989). Catastrophic interference in connectionist networks: the sequential learning problem. In G. H. Bower, *The psychology of learning and motivation* (pp. 109–164). vol. 24. New York: Academic Press.

Moody, J. & Darken, C. (1988). Learning with localized receptive fields. In *Proceedings of the 1988 Connectionist Models Summer School* (pp. 133–143). San Mateo: Morgan Kaufmann.

Murata, N., Müller, K.-R., Ziehe, A. & Amari, S. (1997). Adaptive on-line learning in changing environments. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS 9)* (pp.599–604). Cambridge, MA: MIT Press.

Obradovic, D. (1996). On-line training of recurrent neural networks with continuous topology adaptation. *IEEE Transactions on Neural Networks*, *7*, 222–228.

Park, J., & Sandberg, I. W. (1993). Universal approximation using radial-basis-function networks. *Neural Computation*, *5*, 305–316.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, *3*, 213–225.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, *247*, 978–982.

Prechelt, L. (1994). PROBEN1 a set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, (anonymous FTP: on ftp.ira.uka.de/pub/uni-karlsruhe/papers/techreports/1994/1994-21.ps.gz)

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, *97*, 285–308.

Robbins, H., & Monro, S. (1951). A stochastic approximation model. *Annual Math. Stat.*, *22*, 400–407.

Roy, A., Govil, S., & Miranda, R. (1997). A neural-network learning theory and a polynomial time RBF algorithm. *IEEE Transactions on Neural Networks*, *8*, 1301–1313.

Rumelhart, D. E. Hinton, G. E. & Williams, R. J. (1986). Learning internal

representations by error propagation. In *Parallel distributed processing* (pp. 318–362). Cambridge, MA: MIT Press.

Shadafan, R. S., & Niranjan, M. (1994). A dynamic neural network architecture by sequential partitioning of the input space. *Neural Computation*, *6*, 1202–1223.

Sompolinsky, H., Barkai, N. & Seung, H. S. (1995). On-line learning of dichotomies: algorithms and learning curves. In *Neural networks: the statistical mechanics perspective* (pp. 105–130). Singapore: World Scientific.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks*, *3*, 109–118.

Whitehead, B. A., & Choate, T. D. (1994). Evolving space-filling curves to distribute radial basis functions over an input space. *IEEE Transactions on Neural Networks*, *5*, 15–23.

Yingwei, L., Sundararajan, N., & Saratchandran, P. (1998). Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm. *IEEE Transactions on Neural Networks*, *9*, 308–318.