Radial Basis Function Networks 1: Recent Developments in Theory and Applications.
Ed. by R.J. Howlett and L.C. Jain. Studies in fuzziness and soft computing; vol. 66. Heidelberg, New York: Physica Verlag , Chapter 9: 219-251, 2001.

RBF learning in a non-stationary environment: the stability-plasticity dilemma

Fred H. Hamker

Abstract This chapter focuses on learning with RBF networks in a non-stationary environment. A non-stationary environment demands a neural network to continuously learn. More difficult than following the change is the ability of learning new patterns without forgetting old prototype patterns, also termed as the stability-plasticity dilemma. A local representation and the ability to grow are important prerequisites to face the stability-plasticity dilemma, but not sufficient. Thus, in this contribution a growing RBF network is proposed that learns its number of nodes needed to solve the current task and dynamically adapts the learning rate of each node separately. As shown in several simulations, the RBF network possesses the major characteristics needed to cope with the stability-plasticity dilemma.

1 Introduction

Learning describes the mechanism by which a system obtains and adapts a coherence between the environment and the task. In the past many learning methods have been proposed which also apply to RBF networks. Table 1 aims to bring some order into the terminology, especially from the viewpoint of the stability-plasticity dilemma. Classical RBF learning has dealt with a stationary environment. RBF kernels are placed into the weight space by several deterministic or batch learning procedures, e.g. based on an optimization of an optimal function. The necessity for stochastic on-line learning, in which the parameters of the network are updated only depending on the current sample, arises if not all of the training patterns are available all the time [12]. Generally, a stationary random process is assumed. Interesting real world applications for RBF networks turn up in a non-stationary environment where continuous learning is required. Especially in extension to adaptive on-line learning, where the network tracks changes over time, sequential learning with a RBF network seems to be advantageous, because of its local representation. Sequential learning addresses the ability of repeatedly training a network with new data, without destroying the old prototype patterns. This can be done either by one or by several training samples. Also life-long learning emphasizes learning throughout the entire lifetime without catastrophic interference. Like adaptive on-line learning, the network is faced with a continuous stream of pattern, but previously learned knowledge should be preserved, if it does not contradict to the current task. This demand immediately meets the stability-plasticity dilemma [20]. In some cases the term life-long learning is also used to address task independent learning.

Table 1. Categorization of different learning tasks with regard to the stability-plasticity dilemma.

Environment	set of samples	each sample	
		adapt	adapt and preserve
stationary	incremental		—
	batch or off-line	stochastic on-line	
non-stationary	continuous		
	sequential	adaptive on-line	life long or sequential

1.1 Previous approaches

Learning in artificial neural networks is inevitably connected with forgetting. Later input patterns tend to wash away prior knowledge [20]. While a gradual interference is unavoidable, the sudden and complete erasure of previously well learned pattern – a catastrophic interference – severely limits life-long learning. In distinction to networks with a global or distributed representation of knowledge, like a Multi-Layer-Perceptron, the local representation of Radial Basis Function (RBF) networks is well suited to cope with the problem of forgetting. Nevertheless, they have a fixed number of nodes which has to be determined by the designer. In growing RBF networks 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.

There have been numerous attempts to insert new nodes in RBF networks during learning [36] [43] [11] [15] [40] [35] [27] [38]. The most common insertion or splitting criteria are i) the insertion of a node at the location of the node with the largest error counter [15] [27], ii) the insertion of a new node at the location of the node with the purest class counter [27], iii) the insertion of a node at the location of a new input vector, if it is not covered by the existing activation functions [36] [2] [3] [43] [35] [38]. The recently published M-RAN algorithm [44] [41] uses a combination of novelty in the input and output space and a threshold that is compared with the recent error. Although they used the term sequential learning, they have not addressed changing environments or the stability-plasticity dilemma. Most of the mentioned algorithms aim at an optimal design of the network as a result of an incremental learning process. Only a few focus on adaptive on-line learning [35] and continuous learning [42].

Nevertheless, all these algorithms disregard the stability-plasticity dilemma, if they are used in continuous learning tasks – unlike sometimes claimed, a growing network and a local representation is not sufficient. Why? What makes the difference in learning samples from a non-stationary environment compared to a stationary environment? First of all, possible discrete overlaps of decision areas in finite data sets turn into continuous overlaps, and non-separable areas may emerge, which can not be solved without error. Furthermore, the decision boundaries change over time. Learning in such an environment requires an incremental network that avoids a catastrophic allocation of new nodes. A growing RBF network, in which the insertion depends on the overall error on the current task, has to rely on a maximal number of nodes or on a minimal overall error. Neither of them can be determined with certainty in advance.

Other non-RBF approaches comprise ART-like networks [19] [9] [10] [30] which allow learning only if the pattern matches the stored prototype. While these networks are robust in unsupervised learning, error-driven learning can in case of overlapping decision regions result in a catastrophic allocation of new nodes, if in the course of the Inter-ART-Reset in ARTMAP no appropriate prototype is detected. Again also these networks have not been shown to cope with the hard problems of the stability-plasticity dilemma, such as continuous overlaps of different classes.

1.2 Proposed approach

From the latter arguments it became clear that a better compromise is to dynamically learn how many nodes are needed for an appropriate solution without completely freezing the network and without prescribing a minimal error. Thus, referred to the stability-plasticity dilemma, the essential mechanisms for growing RBF networks are the insertion/deletion of nodes and the learning rate.

1.2.1 Insertion of nodes

A general problem in learning is the bias-variance dilemma [18], where the bias is a measure in how far the average mapping function resembles 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.

In life-long learning tasks growing is an important feature to decrease the error of the task and to adapt to changing environments while preserving old prototype patterns. But for two reasons insertion has to be stopped: to prohibit a permanent increase in the number of nodes in overlapping decision areas, where the task can not be solved and to avoid overfitting. The approach preferred here, is to evaluate the previous insertion by the observation of the error. Because each insertion influences the local behaviour the observed error should also be a local measurement and not the average error on the task. Such an insertion-evaluation cycle allows a local optimization, but decreases the ability to allocate new nodes if previous insertions were not successful.

1.2.2 Adaptive learning parameters

As common procedure to minimize the bias in stochastic on-line learning the learning rate is slowly annealed to zero. 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 in adaptive on-line learning 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 is termed as learning of a learning rule [1].

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

Theoretical considerations of finding the optimal learning rate often assume a global scale factor for all weight-adjustments that depends on the past errors (e.g. [39] [32]). 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 on the chosen neural network, is derived by minimizing a misadjustment, i.e. a small bias and a small variance [25]. For practical purposes the bias and the variance is estimated 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 within the distribution in the input space. To address the stability-plasticity dilemma individual adaptive learning rates for each node are more suitable. One approach is to adapt the level of the learning rate according to the ratio between two local error counters with a different time constant, because this guarantees an asymptotic decrease in case of a local stationary distribution and an increase if the changing environment leads to new local errors.

1.2.3 Framework

The above proposals are implemented within the framework of the Cell Structures [14] [15] [16] [4] [7], because they already support the idea of local error counters. Cell Structures cluster the input space like typical RBF networks, but the nodes are organized within a graph in which the location of the centers and the connecting edges can be updated on-line. It was shown that a competitive Hebbian learning rule enables vector quantizisers to learn perfectly topology preserving mappings [34]. Utilizing this neighborhood relation leads to a cooperative training and performs a similarity regularization which can improve training performance as indicated by excellent results in incremental learning tasks [15] [4] [5] [24] and adaptive on-line learning [17].

The Cell Structures were designed to operate within different learning regimes such as self-organizing, error-driven (or supervised), and reinforcement learning. In their original formulation involving competitive Hebbian learning, all Cell Structures realize unsupervised learning. For supervised learning the output weights updated according to a stochastic gradient descent (least mean square rule, delta rule). They have also been used with reinforcement learning [6].

In extension to an exclusively unsupervised on-line adaptation of the input weights based on Self-Organizing Maps [29] and Neural Gas [33], but without a decay of learning parameters, an error-modulated learning [2] and a more sophisticated gradient-based learning [5] have been suggested. Both introduce an error dependency for the adaptation of centers similar to the one utilized in other RBF algorithms [28].

So far, the major drawback of the 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 equal the input probability density in the unsupervised adaptation case and to equal the error probability density in the supervised adaptation case [21]. A predetermined number of nodes, a dependence on the overall error or on the 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 [17] (high plasticity) or by freezing the amount of nodes and allowing only minor changes in the adaptation of the weights (high stability). This contribution extends the Cell Structures to learn how many nodes are needed for an appropriate solution and how to organize insertion and deletion of nodes in order to tackle the stability-plasticity dilemma.

2 The algorithm

2.1 Structure

The extended Cell Structures, called Life-long Learning Cell Structures (LLCS), perform in their representation layer a vector quantization and consist of nodes or prototypes (Fig. 1). The neighborhood relationship of the nodes is defined by an undirected graph G [34]. 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.

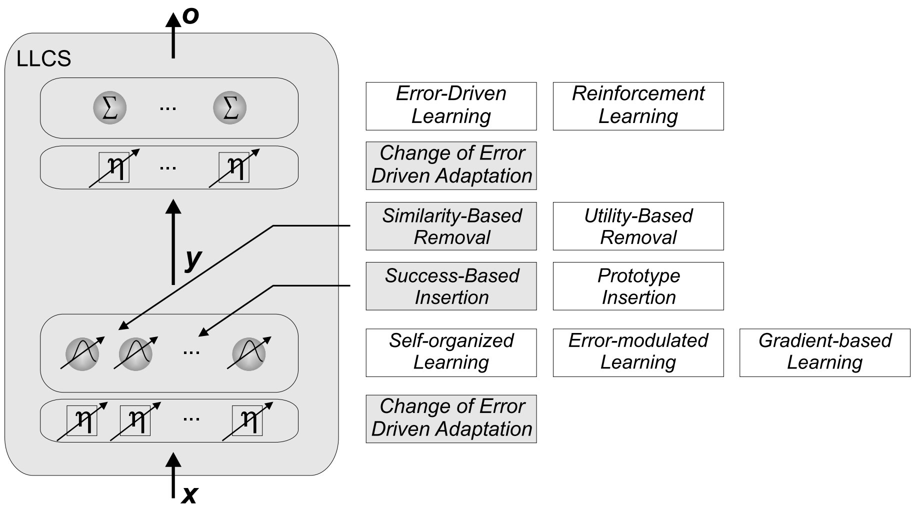


Figure 1. The representation layer composed of Gaussian activation functions performs an adjustable nonlinear transformation of the pattern x. The output layer assigns each activation distribution within the representation layer continuously to a class. By inserting new nodes in regions, which lead to high errors the overall performance is increased. The gray boxes indicate the extension of the Cell Structures for life-long learning in a non-stationary environment.

2.2 Variables of each node

Each node i has a few variables, that regulate learning and insertion of the 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: $$\sigma_i = \frac{1}{\|N_i\|} \sum_{j \in N_i} \|w_i - w_j\| \qquad (2)$$ To avoid abrupt changes, σ_i can be implemented by a moving average.
τ_{Si}	Short-term error counter. The estimate of the average short-term error is adapted according to the time-constant T_S.
τ_{Li}	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.
τ_{Ii}	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 average long-term error τ_L of those two nodes, between which the new one is inserted. Each of 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.

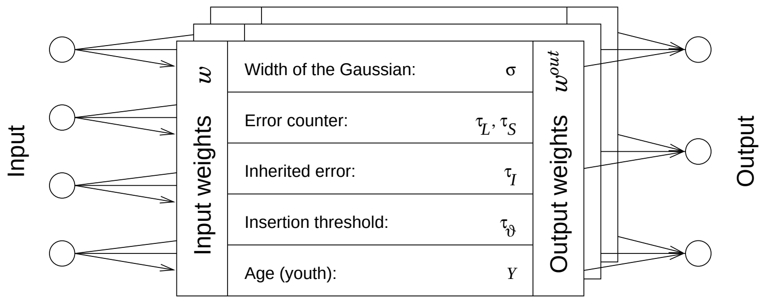

Input
Input weights
w
Width of the Gaussian: σ
Error counter: τ_L, τ_S
Inherited error: τ_I
Insertion threshold: τ_ϑ
Age (youth): Y
Output weights
w^{out}
Output

Figure 2. A RBF 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.

2.3 Adaptation of the representation layer

Only the unit and its neighbours that best match the input pattern are allowed to learn. To locate the best matching unit b, calculate for all nodes i the Euclidian distance d_i of the input pattern $x \in \mathbb{R}^n$ to the weight vector $w_i \in \mathbb{R}^n$.

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

There are many reasons why the learning rate should be adaptive. If the learning rate is held constant: a learning rate chosen too high will corrupt previously learned knowledge and disturb the stability of the output weights, a learning rate chosen too low does not allow the learner to follow a changing environment. Because Cell Structures provide a local processing strategy, a uniform learning rate does not make sense. Each node should hold its individual adaptive learning rate η^i. The weight change is performed by moving the prototype b and its neighbors $c \in N_b$ into the direction of the last training pattern.

$$\triangle w_b = \eta^i_b \cdot (x - w_b); \quad \triangle w_c = \eta^i_c \cdot (x - w_c) \qquad \forall\, c \in N_b \tag{4}$$

To estimate the appropriate learning rate two error counters with different time-constants detect relevant changes in the input probability density, expressed by the quality measure for learning B^L of the best-matching node b and its neighbors $c \in N_b$.

$$B^L_{(b/c)} = \frac{\tau_{S(b/c)} + 1}{\tau_{L(b/c)} + 1} \qquad \forall\, c \in N_b \tag{5}$$

Learning of a node is dependent on its location within the BY-diagram (Fig. 3). The learning rate η^i allows an adaptation of the weights if the nodes are either new or if temporal changes of the error occur. It is determined for the best node b and its neighbors c by the base learning rate of the winner η_b, learning rate of the neighbors η_n, modulated by a localized network based input adaptation term $\alpha^i_{(b/c)}$, which consists of quality measure for learning B^L, the age Y and a pre-defined input adaptation threshold ϑ^i_L

$$\eta^i_{(b/c)} = \begin{cases} 0 & \text{if} \quad \alpha^i_{(b/c)} < 0 \\ \eta_{(b/n)} & \text{if} \quad \alpha^i_{(b/c)} > 1 \\ \alpha^i_{(b/c)} \cdot \eta_{(b/n)} & \text{else} \end{cases}$$
$$\text{with} \quad \alpha^i_{(b/c)} = \frac{B^L_{(b/c)}}{1 + \vartheta^i_L} + Y_{(b/c)} - 1 \tag{6}$$

Within the 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 [7].

A newly inserted node always starts on the right side within the BY-diagram (Fig. 3). The value of the quality measure for learning B^L depends on the nodes, which initiated this insertion, because their counters are used for the initialization. In case of a changing environment, the short term error can increase faster than the long term error which leads to a higher value of B^L. In the opposite case (insertion decreases the error) learning is reduced. If insertion takes place in regions of the input space where decision regions overlap, the quality measure for learning reaches $B^L \approx 1$ and learning is reduced with increasing age (decreasing Y) of the node. The gradient of this cooling process is exponential and dependent 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

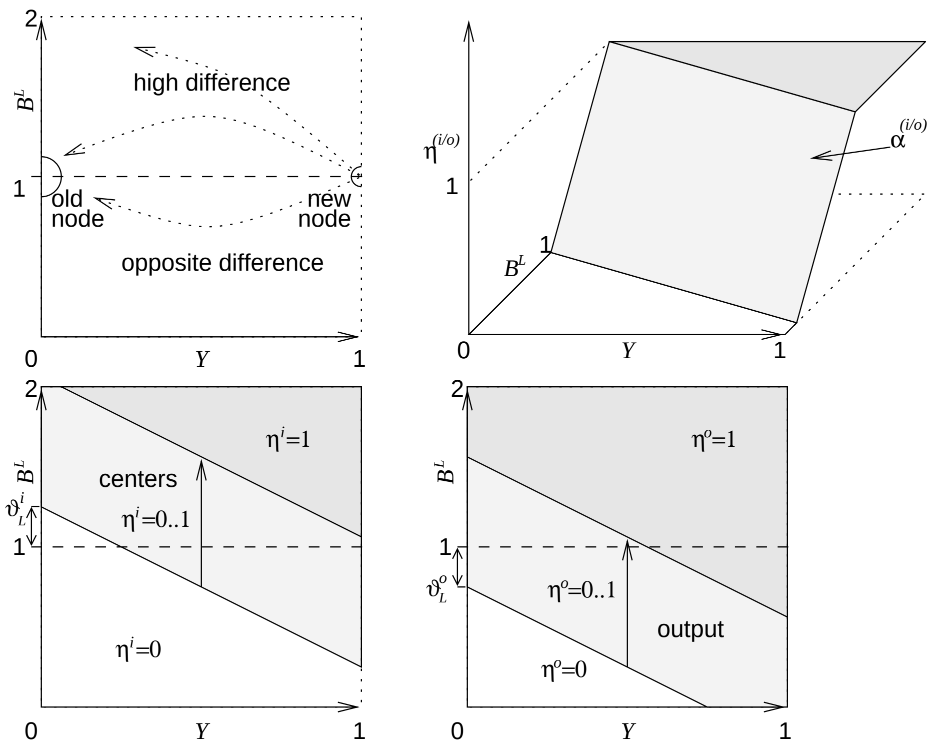


Figure 3. The figure shows the learning rate η in dependence of the quality measure for learning B^L, the age Y and the input/output adaptation threshold $\vartheta_L^{i/o}$. Top left: BY-Diagram to illustrate typical states of the nodes. Top right: Illustration of eq. (6). Bottom left: Learning of the centers and the influence of the user defined input adaptation threshold ϑ_L^i. Bottom right: Learning of the output weights and the influence of the user defined output adaptation threshold ϑ_L^o.

the nodes to reach the state $Y \approx 0; B^L \approx 1$. Besides, suboptimal states are prevented by a further insertion until the local error is not lowered any more.

2.4 Adaptation of the output layer

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

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

Similar as in the representation layer the local output learning rates η^o are determined by the quality measure B^L, the age Y, the output adaptation rate η_o and the output adaptation threshold ϑ^o_L.

$$\eta^o_i = \begin{cases} 0 & \text{if} \quad \alpha^o_i < 0 \\ \eta_o & \text{if} \quad \alpha^o_i > 1 \\ \alpha^o_i \cdot \eta_o & \text{else} \end{cases}$$
$$\text{with} \quad \alpha^o_i = \frac{B^L_i}{1 + \vartheta^o_L} + Y_i - 1 \quad \forall\, i \in G \tag{8}$$

Finally, the weights of the nodes j of the output layer are adapted.

$$\triangle w_{ji} = \eta^o_i\,(\zeta_j - o_j)\,y_i \qquad \forall\, j \in \{1 \ldots m\},\ \forall\, i \in G \tag{9}$$

But only those, which show a sufficient high activation y_i in the representation layer, calculated with a Gaussian function.

$$y_i = e^{-\frac{\|x - w_i\|^2}{\sigma_i^2}} \qquad \forall i \in G \tag{10}$$

2.5 Insertion and deletion of nodes in the representation layer

According to the concept of the local error based insertion criterion the nodes in the representation layer compete to determine the node with the highest similarity to the pattern. By maintaining a local error counter for each node in the representation layer, new nodes are inserted next to input patterns which lead to high errors. This substance of the Cell Structures, the error-driven insertion, has its origin in the GCS. It ensures that the resources are spread over a period of presented patterns, which leads to a better exploitation of the overall resources. Additional criteria, e.g. a prototype-insertion [2], can be used simultaneously to speed up on-line learning.

How does the network learn whether a further insertion of nodes is useful to solve the task? After a number of learning steps, the average error of a node is compared to the error at the moment of the last insertion (Fig. 4). If this error is greater or equal, the insertion was not successful and a local insertion threshold attached to each node 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 has to be decreased by the change of the error. The chance of insertion is

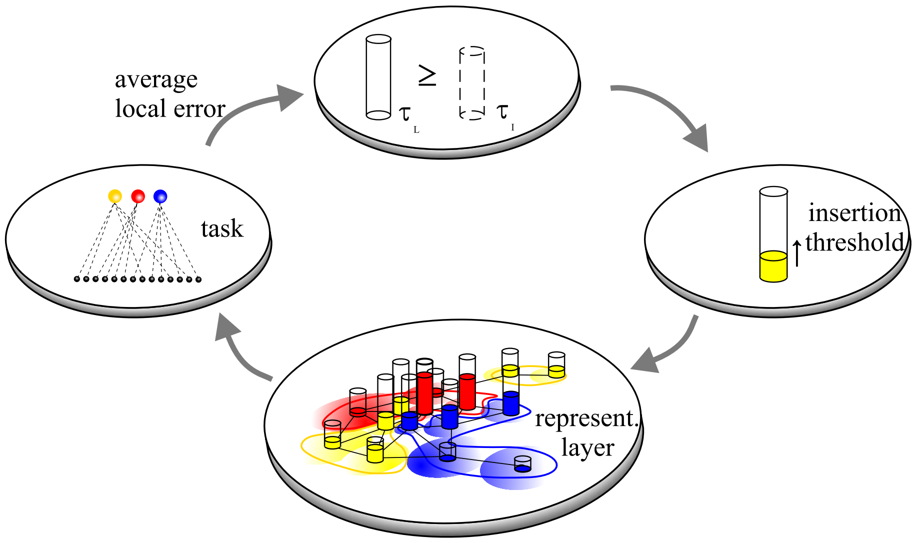


Figure 4. Insertion-evaluation cycle. By the comparison of the current local error with the previous error the last insertion is evaluated.

investigated after each $T_{ins} = \lambda \cdot n_N$ steps by determining the quality measure for insertion B^I considering the insertion tolerance ϑ_{ins}.

$$B_i^I = \tau_{Li} - \tau_{\vartheta i} \cdot (1 + \vartheta_{ins}) \qquad \forall\, i \in G \tag{11}$$

But not only the distance between the long-term error τ_L and the insertion threshold τ_ϑ is decisive. An insertion is only allowed, if a node q is found with a maximal but positive insertion criterion K_{ins}, which also considers the age Y of the node. In this case a new node is inserted between q and f, which is determined among the neighbours of q.

$$\begin{aligned} 0 < K_{ins,q} &= \max_{i \in G} (K_{ins,i}); \quad K_{ins,i} = B_i^I - Y_i \quad \forall\, i \in G \\ B_f^I &= \max_{c \in N_q} (B_c^I); \end{aligned} \tag{12}$$

This criterion is only error based, which supports the acquisition of nodes in regions with high errors independent of the input probability density. Insertion means the edge between q and f is deleted and a new node r is inserted, and connected with q and f. The weights w_r, w_r^{out} as well as the counters τ_{Sr}, τ_{Lr}, and $\tau_{\vartheta r}$ are determined by the arithmetical average of the corresponding weights and error counters of q and f.

The last insertion is evaluated by comparing the long-term error τ_L of q, f and r with the inherited error τ_I lowered by an insertion tolerance ϑ_{ins}.

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

If τ_L exceeds this memory term, the last insertion was not successful, and the insertion threshold has to be adapted.

$$\begin{gathered} \tau_{\vartheta i} := \tau_{\vartheta i} + \eta_{\vartheta} \cdot (\tau_{Li} - \tau_{\vartheta i} \cdot (1 - \vartheta_{ins})) \\ \forall\, i \in \{k | \tau_{Lk} > \tau_{Ik} \cdot (1 - \vartheta_{ins}); q, f, r\} \end{gathered} \tag{14}$$

Finally, by the assignment of the present long-term error to the inherited error τ_I of q, f and r, the memory is updated.

$$\tau_{Ii} = \tau_{Li} \qquad \forall\, i \in \{q, r, f\} \tag{15}$$

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

Looking back at the previous discussion on the insertion of nodes, an incremental neural network can not know in advance whether a further insertion reveals a subtle distribution or turns out as a waste of resources. This is closely related to the bias-variance dilemma. An insertion 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 and variance. 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 fewer insertions are possible until the insertion threshold reaches the long-term error. This criterion detects the decision boundaries between distinctly separated classes but avoids a too low bias in areas with much overlap. The insertion tolerance ϑ_{ins} determines how much the algorithm tolerates a fluctuation of the long-term error without initiating an insertion. It should therefore not be used to address the bias-variance dilemma.

Another criterion acts after an insertion and removes similar nodes. The larger ϑ_{del} the earlier similar nodes are deleted. A node d is only deleted, if it has a minimal age ϑ_{delY}, a sufficient stabilization ϑ_{delB^L}, a minimal number of edges and if its criterion K is lower than the deletion threshold ϑ_{del}.

$$\begin{aligned} &\vartheta_{del} > K_{del,d} = \min_{i \in G} \left(K_{del,i} \right) \wedge \\ &\|N_d\| \geq 2 \ \wedge \ Y_d < \vartheta_{delY} \ \wedge \ B_d^L < \vartheta_{delB^L} \end{aligned} \tag{16}$$

with

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

the local similarity of the input weights:

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

the average similarity of the input weights:

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

and the local similarity of the output weights:

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

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

The long-term error counter τ_{Lb} and the short-term error counter τ_{Sb} for the winner b are updated as a moving average.

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

The age Y_b of the best-matching node b is simply decreased.

$$Y_b := e^{-\frac{1}{T_Y}} \cdot Y_b \tag{22}$$

A prerequisite for the flexibility of insertion in changing environments is the decrease of the insertion threshold $\tau_{\vartheta b}$, if the distribution of the error changes.

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

with

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

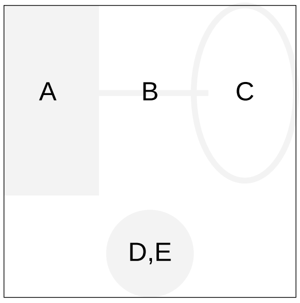
The edges of the graph are continuously updated. According to the Hebbian learning rule the age of all edges emanating from b are increased by one and the age of the edge between b and the second best s is set to zero. The second best node is defined as:

$$d_s = \min_{i \in G, i \neq b}(d_i); \qquad d_i = \|x - w_i\| \quad \forall\, i \in G \tag{25}$$

If no edge between b and s exists, a new one is created. All edges older than ϑ_{age} and all nodes without an edge are removed.

2.7 Parameter discussion

The algorithm expects the specification of several parameters. The major parameters that concern the insertion and deletion, i.e. the size of the network, are the learning rate of the insertion threshold η_ϑ and the deletion threshold ϑ_{del}. The sensitivity to temporal changes of the environment is adjusted by the relation of the time constants of the short-term error and the long-term error T_S/T_L. The insertion threshold ϑ_{ins} defines the sensitivity to changes of the long-term error regarding an insertion of nodes. Other parameters can be regarded as constants of the algorithm. This insensitivity to parameter settings is a general feature of the Cell Structures as indicated by a benchmark [24].



	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

Figure 5. Changing environment composed of five areas ($A - E$) and three classes. The environment changes from 1-6 each 20000 steps. Simulation parameters: $\eta_b = 0.1$, $\eta_n = 0.01$, $\eta_o = 0.15$, $\eta_\vartheta = 0.5$, $T_S = 20$, $T_L = T_Y = T_\vartheta = 100$, $\lambda = 10$, $\vartheta_{age} = 50$, $\vartheta_L^i = 0.05$, $\vartheta_L^o = -0.05$, $\vartheta_{ins} = 0.1$, $\vartheta_{del} = 0.05$, $\vartheta_{delY} = 0.01$, $\vartheta_{delB^L} = 0.01$.

3 Illustration with artificial data sets

To illustrate the function of the RBF learning in a non-stationary environment, the behavior of the network on a two-dimensional data set is observed (Fig. 5). It is shown that even critical overlaps do not lead to a permanent insertion, while the network is not frozen.

In the first 20000 steps the input contains an awful overlap in the circular area which causes a high error. Thus, initially the internal states of the nodes responsible for the overlapping area show a high long-term error. From left to right, two plots in Fig. 6 show the states in each environment, the first after each 500 steps and the second after each 20000. As the learning parameter of the input weights expresses, the network is extremely plastic. Nevertheless after 20000 steps, the algorithm has learned by increasing its insertion threshold, that a further insertion does not improve the squared error and stabilizes, as can be seen from the learning parameters and the number of nodes.

Now the environment changes, new errors occur and the algorithm tries to minimize them by changing its weights and inserting new nodes. Although the environment 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 40000 steps it has adopted to the environment such that no further learning is needed.

If the probability changes to zero in some regions, like in the environment from 40000 to 60000 steps, those remaining nodes, often called ”dead nodes”, play a major role for the stability-plasticity dilemma (Fig. 7). 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 stabilizes after 50000 steps.

In the environment from 60000 to 80000 steps, most of the nodes remain at their positions. The repeated appearance of area A does not raise the error – the knowledge was completely preserved. Since the environment shows no overlaps the error decreases to zero.

In the environment from 80000 to 100000 steps, the patterns from the circular area change from class two to class three (Fig. 8). This change of the environment illustrates impressively the localized definition of the stability and plasticity. The network at 80500 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 their leaning rate. It turns out, that the new nodes all cover the same class and most of them were again deleted. Once more the network stabilizes.

Even serious changes in the environment from 100000 to 120000 are tolerated. The invertation of the occurrence of patterns in area A, B, and C does not affect the position of the centers. The overlap of area D and E raises the error and the network inserts new nodes but stabilizes again.

Summarizing, the algorithm is able to cope with all changing environments, like overlaps, never seen inputs and temporarily not appearing patterns.

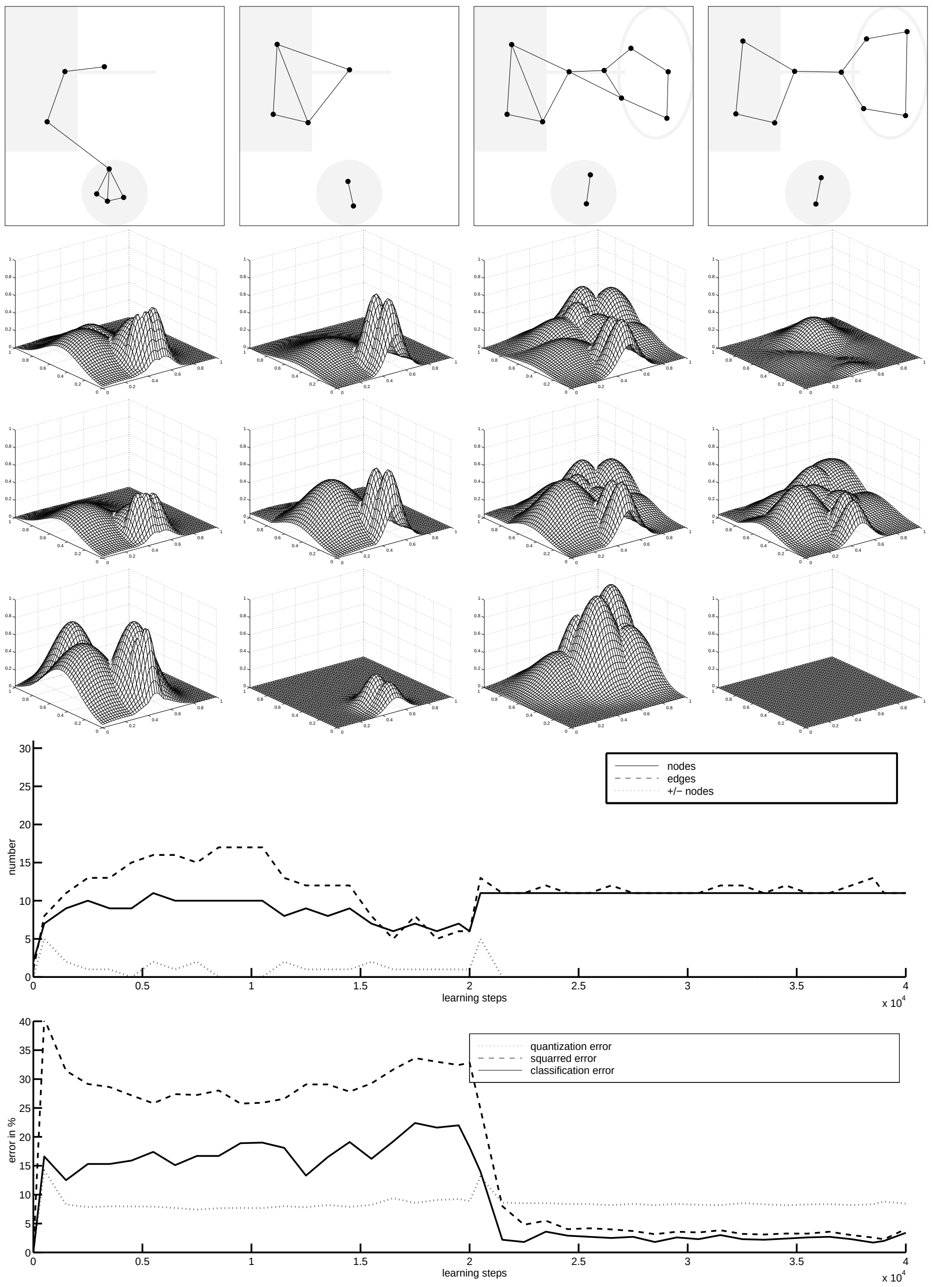


Figure 6. Internal states of the RBF nodes in environment 1 and 2. From the top to the bottom, the input weights w_i, the long-term error τ_{Li}, the insertion threshold $\tau_{\vartheta i}$, the adaptation term of the input learning rate α_i^i are presented.

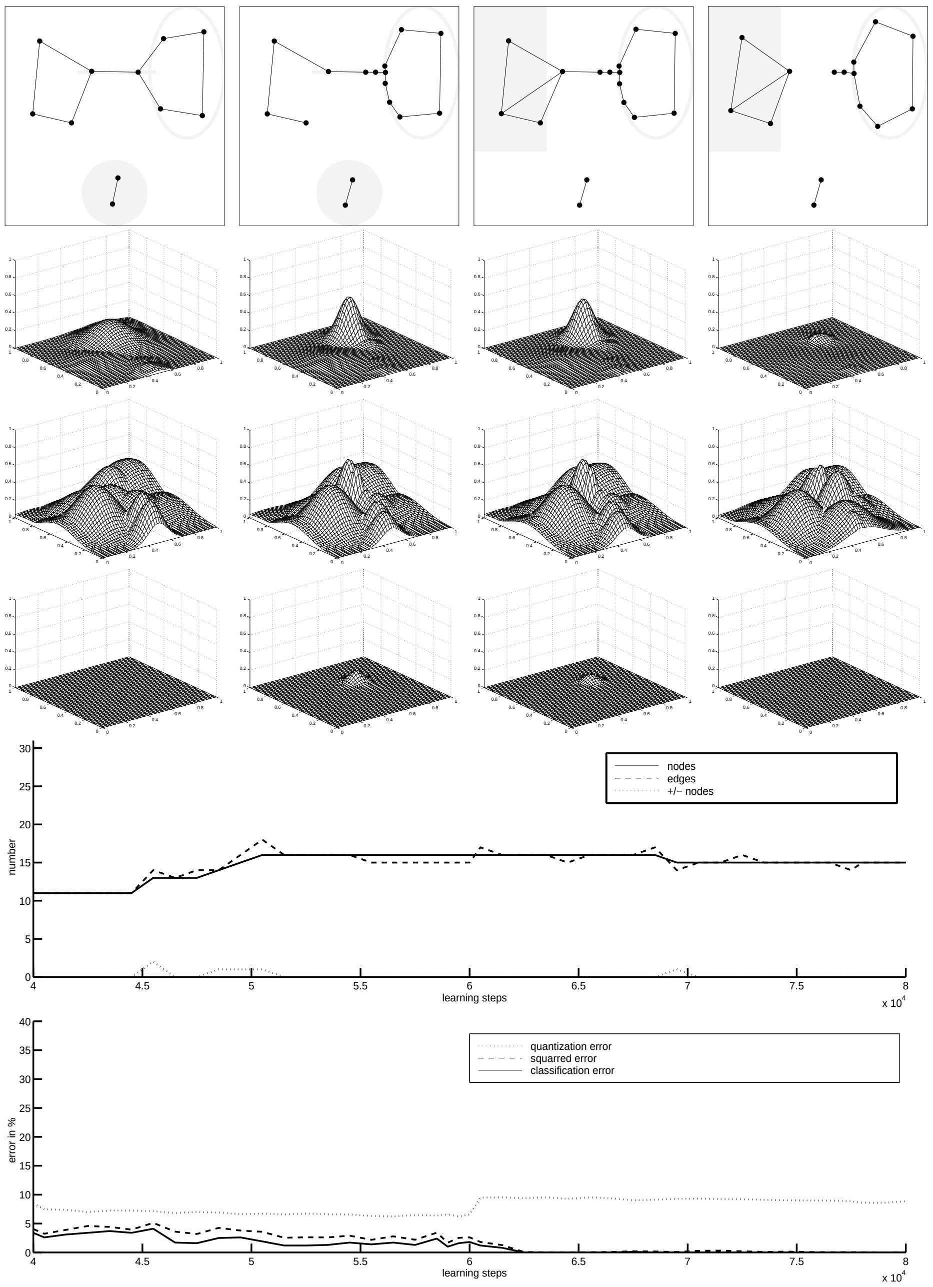


Figure 7. Internal states of the RBF nodes in environment 3 and 4. From the top to the bottom, the input weights w_i, the long-term error τ_{Li}, the insertion threshold $\tau_{\vartheta i}$, the adaptation term of the input learning rate α_i^i are presented.

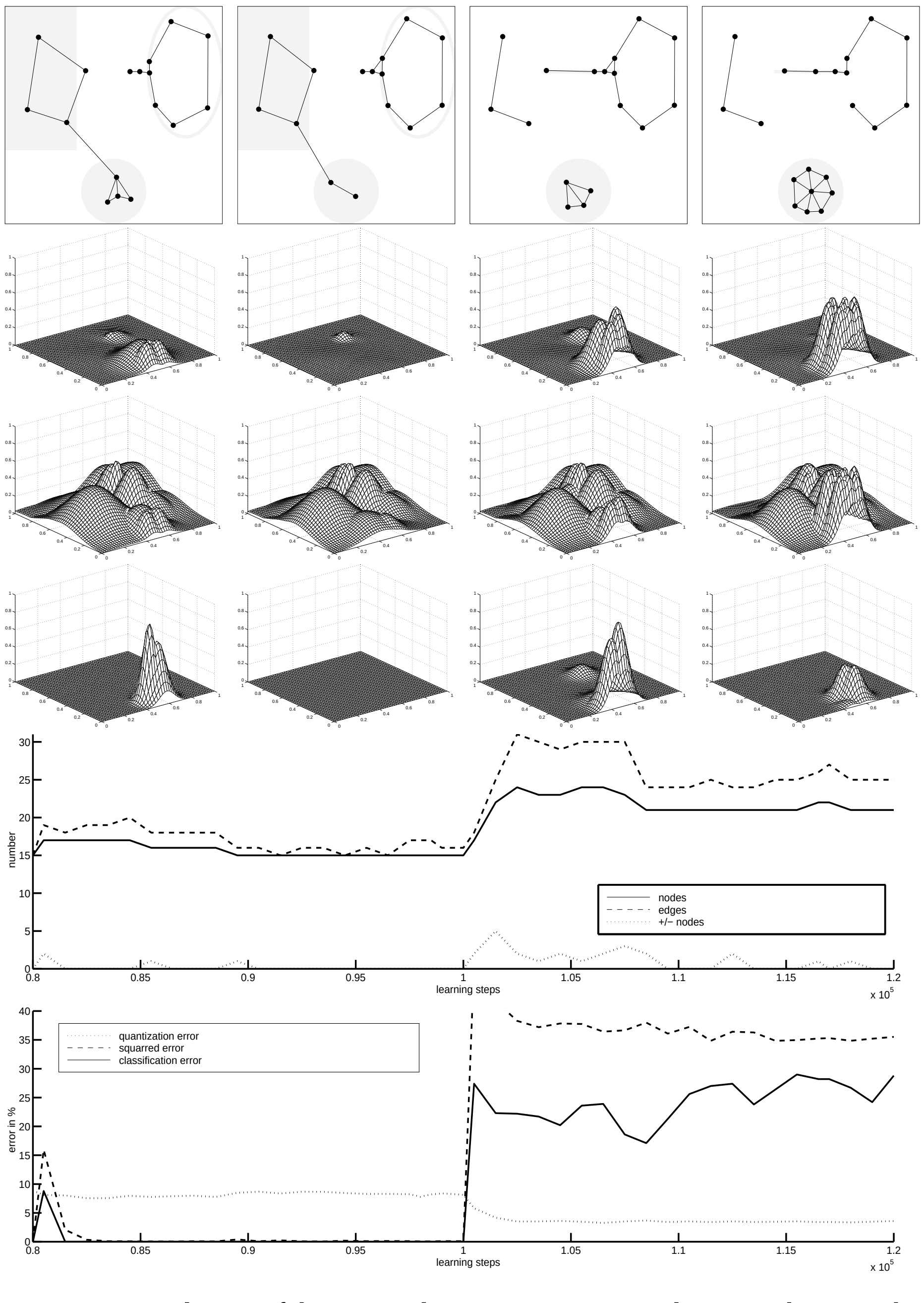


Figure 8. Internal states of the RBF nodes in environment 5 and 6. From the top to the bottom, the input weights w_i, the long-term error τ_{Li}, the insertion threshold $\tau_{\vartheta i}$, the adaptation term of the input learning rate α_i^i are presented.

4 Evaluation using real data sets

4.1 Performance measures

Relevant issues for the application of the LLCS deal with non-stationary input probability distributions. Nevertheless, it was shown that the network automatically stops the insertion of further nodes on real benchmark data with a stationary probability distribution and achieves a performance which is as good as those of the Cell Structures selected by cross-validation [23] [24]. For the further evaluation of learning in changing

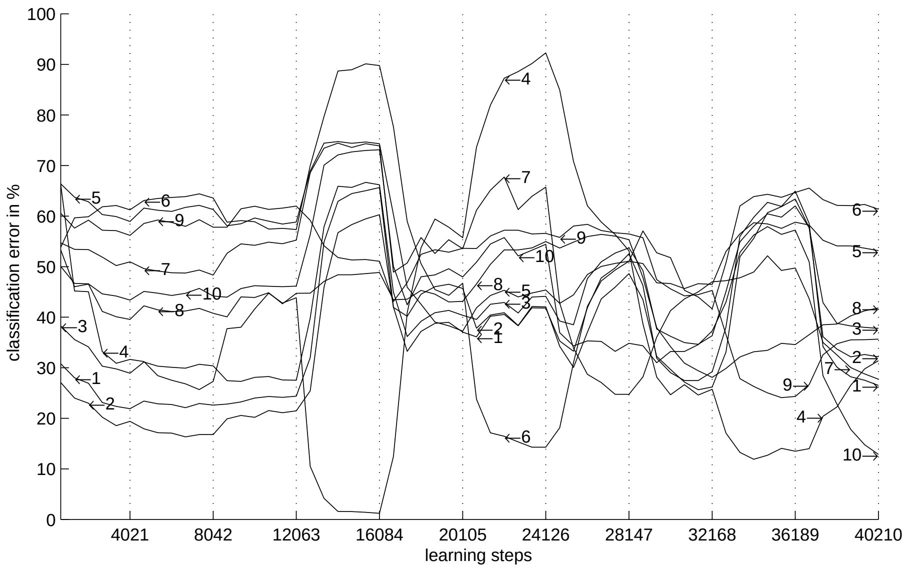


Figure 9. Average classification error of the environments 1-10 gained from 10 different runs by changing the environment after each 4021 steps. Parameters: $\eta_\vartheta = 0.1$, $\vartheta_{del} = 0.2$, $\eta_b = 0.8$, $\eta_n = 0.01$, $\eta_o = 0.01$, $T_S = 100$, $T_L = T_Y = T_\vartheta = 200$, $\lambda = 100$, $\vartheta_{age} = 60$, $\vartheta_{ins} = 0.4$, $\vartheta_L^i = 0.1$, $\vartheta_L^o = -0.05$, $\vartheta_{delY} = 0.01$, $\vartheta_{delB^L} = 0.01$.

environments a data set with strongly overlapping decision areas was designed. It consists of 10 environments with 29 features and four classes. Each environment was build from four images which contain four different materials (classes), like journals, cardboard, newspaper and others, recorded under different lightning conditions. The patterns are gained from color-histograms of tiles sized 32x32 pixels. For details about the data and the feature extraction see [22]. One after another the data from an environment is only once presented (on-line learning). But even the data from the environment is clumped into 402 blocks with 10 samples belonging to the same class, which makes learning more difficult and more realistic to natural environments, which are structured in space and/or time. The output of the network on all data sets are recorded in parallel (Fig. 9). This means the impact of training a particular data set on the performance on any other data set can be analyzed. According to the correspondence between different data sets learning in one environment is of advantage to some environments while others suffer from strong overlaps.

If we look at fig. 9 we wonder if the network is stable or not. This raises the question how to measure the stability and plasticity of a neural network.

Plasticity means a neural network is capable to learn new patterns. A high plasticity reveals that the network is not frozen after different data sets were presented. Thus, we have to focus not on the overall error, but on the difference of the performance between the network trained within an environment and the network seen different environments before. For a quantitative evaluation two bounds were defined. The upper bound was gained by training the network only once in an environment. The lower bound was achieved by training the network in the same environment as long as the tested network was trained in different environments. Thus, the number of nodes is similar. Each bound is an average of 10 runs. The error can not fall below the lower bound. Achieving the upper bound is a good performance, an error lower than the upper bound is even better.

Stability is the crucial aspect of the stability-plasticity dilemma. The illustration on artificial data has impressively demonstrated the capability to preserve the prototypes while confronted with new patterns elsewhere from the input space. Nevertheless, if the environment changes, the new situation can be inconsistent with the former situation and the network has to adapt itself. This means it looses some previous knowledge in order to respond accurately. For a quantitative analysis of stability criteria

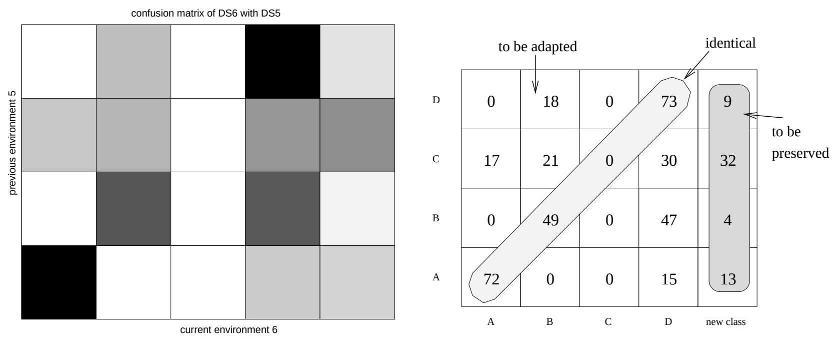


Figure 10. Left: Extract from the confusion matrix of two environments with $\gamma = 0.5$. The cut shows environment 6 and 5. A black rectangle indicates a large value. Right: Evaluation of the confusion matrix. The values on the main diagonal mark the correspondence of the environments. Even if the data from environment 5 was not presented for training, the samples of the main diagonal could be classified correctly while training in environment 6. The patterns of other fields except from the class C_{new} collide with the current environment and are expected to be not preserved according to the plasticity demand. But the amount of pattern from class C_{new} could be expected to be preserved when the network is trained on data from environment 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 $\overline{S_\gamma} = 100\% - \frac{194\% + 58\%}{4} = 37\%$.

like measuring the amount of forgetting [31] [37] and the amount of time to re-learn the patterns [26] are used. The latter offers only little insight into the stability characteristic, because it depends too much on the algorithms learning strategy. On the first sight the amount of forgetting seems to be a useful criterion, but forgetting is a necessary condition of plasticity in overlapping decision areas. An appropriate measure has to define which data has to be preserved and which should be adapted according to the plasticity condition. Prototypes in overlapping decision areas are expected to adapt to the new environment without affecting others, and unseen or new patterns should enlarge the knowledge. Thus, we have to estimate the amount of samples from the previous environment that is similar or new to the data from the current environment. We expect this amount of samples to be classified correctly while training with patterns from the current environment.

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. This concept is extended to estimate the confusion of data from different environments. In order to achieve a more precise criterion all patterns that result in an output activation lower than a threshold $\gamma \in \{0.1, 0.3, 0.5\}$, were declared as a not sufficiently learned patterns with a separate class (new), instead of being assigned to the class with the largest output activation. Using this definition for the class C_{new} we obtain the stability boundary $\overline{S_\gamma}$:

$$\overline{S_\gamma} = 100\% - (\#correspondence + \#new_\gamma)\% \qquad (26)$$

Fig. 10 illustrates this context with the data from environment five and six. 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.

4.2 Analysis of the data

Each environment contains of 4096 samples. Most environments show a strong overlap between their classes (Fig. 11). From the first environment to the third only slow changes, whereas afterwards abrupt changes occur. As can be seen from fig. 11, several environments do not contain all classes, especially environment four contains only patterns of one class that has a strong overlap to different classes in other environments. This is the reason of the large increase of error when data from environment four is presented (Fig. 9). Especially the error of environment six increases strongly because both environments share no common class. Only the performance on environment nine can profit by the training with data of environment four, because both share similar patterns (Fig. 11).

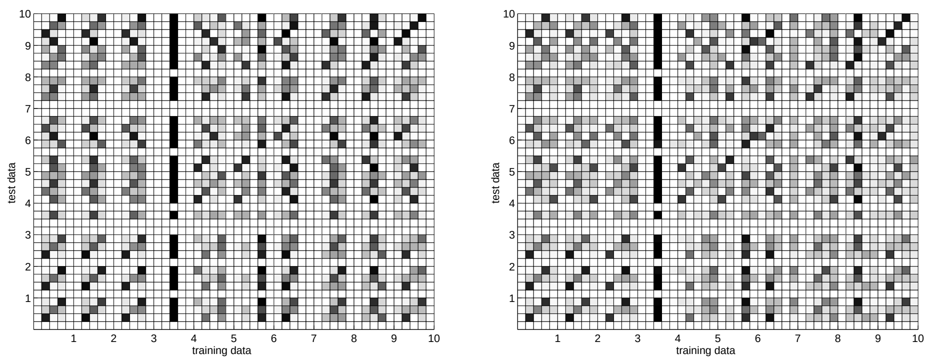


Figure 11. Confusion matrix of all data sets with $\gamma = 0.1$ (left) and $\gamma = 0.5$ (right) after the network was trained for three epochs on the data of an environment. See fig. 10 for the structure of the matrix. Environments with no overlap (high similarity) show a black diagonal within each 4x5 array. Activations at other areas indicate a complete overlap. A larger threshold γ results in a bigger class C_{new}.

4.3 Analysis of stability and plasticity

The course of errors in fig. 12 demonstrates a good plasticity performance. After the environment changes the error decreases and in most cases it falls below the upper bound. Concerning stability, the change from environment 1 to environment 2 is remarkable. Here, the classification error is larger than the bound, but this is based on the short training time and the few number of nodes, which hinder a precise representation of the data. We have to keep in mind, that the bounds were estimated after the training of three epochs. The further course clearly indicates the preservation of old prototype patterns. Even networks with more conservative parameter sets show a convincing stability and plasticity performance, although these networks operate with less than half of the nodes as shown in [23].

5 Conclusion

Learning in changing environments is faced with the catastrophic interference, which mainly occurs in a distributed representation. But also

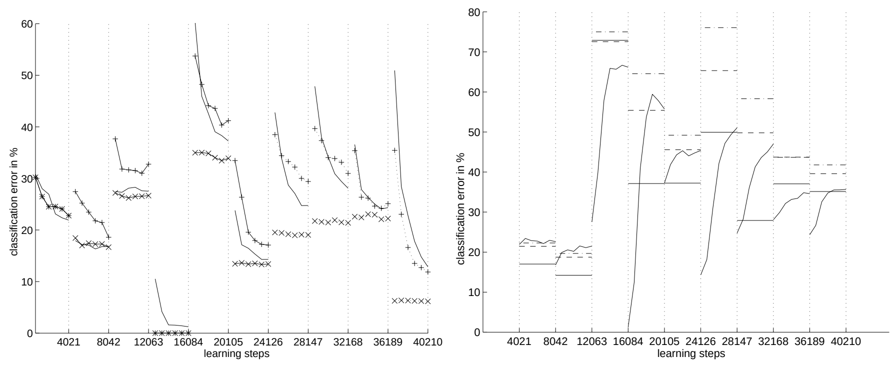


Figure 12. The course of the error in different environments are gained from the recordings of fig. 9. Left: Results of the plasticity analysis. The error of the network is shown when presented a never seen environment (–), compared to the upper bound (+) and the lower bound (x). Right: Results of the stability analysis. The figure shows the stability bounds estimated with $\gamma = 0.1$ ($- \cdot -$), $\gamma = 0.3$ (– –) and $\gamma = 0.5$ (—) compared to the error in the previously trained environment (–), e.g., environment 7 is trained and environment 6 is observed. The error of the first two environments exceeds the estimated bounds. This is due to the low number of nodes created so far. In most cases the error is about or below the lowest bound, which underlines the exceptional stability property of the LLCS. Only in some cases, mostly if the training on the previous data set does not decrease the error onto the level of the reference network, which determines the bound, the error exceeds the lowest bound. But even then the networks still preserve some knowledge.

localist or partly distributed representations suffer from interference, especially if the network does not have sufficient nodes. Thus, different strategies to insert new nodes in incremental neural networks were mentioned. A similarity based insertion as used in ART networks [8] and also in RBF networks [36] [44] has the advantage to increase the number of nodes without an instability concerning insertion, but 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. A different strategy is to insert a new node according to an error based criterion. While a global error based insertion criterion is not useful for life-long learning, because the error is not known in advance and changes over time. A promising strategy is a local error based insertion criterion as used by GCS [14] or also in ARTMAP [9] triggered by the Inter-ART-Reset. But this raises the question how to suppress insertion in overlapping decision areas, where errors occur all the time. Furthermore, learning in changing environments 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.

The evolved RBF network is based on the Cell Structures algorithm [14] [16] [4] [7] and extends it to locally adapt the stability and plasticity – for learning and for insertion. The essential innovation compared to the previous work is the new interpretation 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 in case of 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 proposed algorithm is 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 are 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 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.

Although still much has to be done, RBF 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.

Acknowledgments

The body of this work was done at the Department of Neuroinformatik, Technische Universität Ilmenau (Germany). I thank T. Vesper for his fruitful discussions and for implementing parts of the algorithm in his diploma thesis. As a foundation of this research, the combination of action and perception, a research orientation of Prof. H.-M. Gross is worth mentioning. I would also like to thank D. Surmeli for his helpful comments on a preliminary version of this chapter.

References

[1] Amari, S. (1967) "A theory of adaptive pattern classifiers," *IEEE Transactions on Electronic Computers*, Vol. 16, pp. 299-307.

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

[3] Berthold, M.R. and Diamond, J. (1995), Boosting the performance of RBF networks with dynamic decay adjustment," *Advances in Neural Information Processing Systems (NIPS 7)*. MIT Press, Cambridge, pp. 521-528.

[4] Bruske, J. and Sommer, G. (1995), "Dynamic cell structure learns perfectly topology preserving map," *Neural Computation*, Vol. 7, pp. 845-865.

[5] Bruske, J., Hansen, M., Riehn, L. and Sommer, G. (1996), "Adaptive saccade control of a binocular head with Dynamic Cell Structures," *Proceedings of the International Conference on Artificial Neural Networks*, pp. 215-220.

[6] Bruske, J., Ahrns, I. and Sommer, G. (1998), "An integrated architecture for learning of reative behaviors based on dynamic cell structures," *Robotics and Autonomous Systems*, Vol. 22, pp. 87-102.

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

[8] Carpenter, G.A. and Grossberg, S. (1987), “ART2: Self-organisation of stable category recognition codes for analog input patterns,“ *Applied Optics*, Vol. 26, pp. 4919-4930.

[9] Carpenter, G.A., Grossberg, S. and Reynolds, J.H. (1991), “ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing-neural network,“ *Neural Networks*, Vol. 4, pp. 543-564.

[10] Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H. and Rosen, D.B. (1992), Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps,“ *IEEE Transactions on Neural Networks*, Vol. 3, pp. 698-713.

[11] Chen, Y.Q., Thomas, D.W. and Nixon, M.S. (1994), “Generating-shrinking algorithm for learning arbitrary classification,“ *Neural Networks*, Vol. 7, pp. 1477-1489.

[12] Freeman, J.A.S. and Saad, D. (1997), “On-line learning in radial basis function networks,“ *Neural Computation*, Vol. 9, pp. 1601-1622.

[13] French, R.M. (1999), “Catastrophic forgetting in connectionist networks,“ *Trends in Cognitive Sciences*, Vol. 3, pp. 128-135.

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

[15] Fritzke, B. (1994), “Growing cell structures – a self-organizing network for unsupervised and supervised learning,“ *Neural Networks*, Vol. 7, pp. 1441-1460.

[16] Fritzke, B. (1995), “A growing neural gas network learns topologies,“ *Advances in Neural Information Processing Systems (NIPS 7)*. MIT Press, Cambridge, pp. 625-632.

[17] Fritzke, B. (1997), “A self-organizing network that can follow non-stationary distributions,“ *Proceedings of the International Conference on Artificial Neural Networks*. Springer, pp. 613-618.

[18] Geman, S., Bienenstock, E. and Doursat, R. (1992), “Neural networks and the bias/variance dilema,“ *Neural Computation*, Vol. 4, pp. 1-58.

[19] Grossberg, S. (1976), “Adaptive pattern classification and universal recoding: I.Parallel development and coding of neural feature detectors,“ *Biological Cybernetics*, Vol. 23, pp. 121-134.

[20] Grossberg, S. (1988), “Nonlinear neural networks: Principles, mechanisms, and architectures,“ *Neural Networks*, Vol. 1, pp. 17-61.

[21] Hamker, F.H. and Gross, H.-M. (1997), “Task-based representation in lifelong learning incremental neural networks,“ *VDI Fortschrittberichte*, Reihe 8, Nr. 663, Workshop SOAVE’97 (Ilmenau), pp. 99-108.

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

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

[24] Heinke, D. and Hamker, F.H. (1998), “Comparing Neural Networks: A Benchmark on Growing Neural Gas, Growing Cell Structures, and Fuzzy ARTMAP,“ *IEEE Transactions on Neural Networks*, Vol. 9, pp. 1279-1291.

[25] Heskes, T.M. and Kappen, B. (1993), “On-line learning processes in artificial neural networks,“ *Mathematical Foundations of Neural Networks*, Elsevier Science Publishers, Amsterdam, pp. 199-233.

[26] Hetherington, P. and Seidenberg, M. (1989), “Is there “catastrophic interference“ in connectionist networks?“ *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, LEA, Hillsdale, pp. 26-33.

[27] Karayiannis, N.B. and Mi, G.W. (1997), “Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques,“ *IEEE Transactions on Neural Networks*, Vol. 8, pp. 1492-1506.

[28] Karayiannis, N.B. (1999), “Reformulated radial basis neural networks trained by gradient descent,“ *IEEE Transactions on Neural Networks*, Vol. 10, pp. 657-671.

[29] Kohonen, T. (1982), “Self-organized formation of topologically correct feature maps,“ *Biological Cybernetics*, Vol. 43, pp. 59-69.

[30] Lim, C.P. and Harrison, R.F. (1997), “An incremental adaptive network for on-line supervised learning and probability estimation,“ *Neural Networks*, Vol. 10, pp. 925-939.

[31] McCloskey, M. and Cohen, N. (1989), “Catastrophic interference in connectionist networks: The sequential learning problem,“ *The Psychology of learning and motivation*. Vol. 24, Academic Press, New York, pp. 109-164.

[32] Murata, N., Müller, K.-R., Ziehe, A. and Amari, S. (1997), “Adaptive on-line learning in changing environments,“ *Proceedings of the Conference on Neural Information Processing Systems (Nips 9)*, MIT Press, pp. 599-604.

[33] Martinetz, T.M. and Schulten, K.J. (1991), “A “neural gas“ network learns topologies,“ *Artificial Neural Networks*, volume I, Amsterdam: North Holland, pp. 397-402.

[34] Martinetz, T.M. and Schulten, K.J. (1994), “Topology representing networks,“ *Neural Networks*, Vol. 7, pp. 507-522.

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

[36] Platt, J. (1991), “A resource-allocating network for function interpolation,“ *Neural Computation*, Vol. 3, pp. 213-225.

[37] Ratcliff, R. (1990), “Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions,“ *Psychological Review*, Vol. 97, pp. 285-308.

[38] Roy, A., Govil, S. and Miranda, R. (1997), “A neural-network learning theory and a polynomial time RBF algorithm,“ *IEEE Transactions on Neural Networks*, Vol. 8, pp. 1301-1313.

[39] Sompolinsky, H., Barkai, N. and Seung, H.S. (1995), “On-line learning of dichotomies: algorithms and learning curves,“ *Neural Networks: The Statistical Mechanics Perspective*. World Scientific, Singapore, pp. 105-130.

[40] Shadafan, R.S. and Niranjan, M. (1994), “A dynamic neural network architecture by sequential partitioning of the input space,“ *Neural Computation*, Vol. 6, pp. 1202-1223.

[41] Sundararajan, N., Saratchandran, P. and YingWei, L. (1999), *Radial basis function neural networks with sequential learning : MRAN and its applications*. Series: Progress in neural processing, Vol. 11., World Scientific, Singapore.

[42] Tagscherer, M. (1998), “ICE - an incremental hybrid system for continuous learning,“ *Proceedings of the International Conference on Artificial Neural Networks*, pp. 597-602.

[43] Whitehead, B.A. and Choate, T.D. (1994), “Evolving space-filling curves to distribute radial basis functions over an input space,“ *IEEE Transactions on Neural Networks*, Vol. 5, pp. 15-23.

[44] Yingwei, L., Sundararajan, N. and Saratchandran, P. (1998), “Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm,“ *IEEE Transactions on Neural Networks*, Vol. 9, pp. 308-318.