Task-dependent evolution of modularity in neural networks[1]

Michael Hüsken, Christian Igel, and Marc Toussaint

Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany
Telephone: +49 234 32 25558, Fax: +49 234 32 14209
{huesken,igel,mt}@neuroinformatik.ruhr-uni-bochum.de

Connection Science Vol 14, No 3, 2002, p. 219-229

Abstract. There exist many ideas and assumptions about the development and meaning of modularity in biological and technical neural systems. We empirically study the evolution of connectionist models in the context of modular problems. For this purpose, we define quantitative measures for the degree of modularity and monitor them during evolutionary processes under different constraints. It turns out that the modularity of the problem is reflected by the architecture of adapted systems, although learning can counterbalance some imperfection of the architecture. The demand for *fast* learning systems increases the selective pressure towards modularity.

1 Introduction

The performance of biological as well as technical neural systems crucially depends on their architectures. In case of a feed-forward neural network (NN), architecture may be defined as the underlying graph constituting the number of neurons and the way these neurons are connected. Particularly one property of architectures, *modularity*, has raised a lot of interest among researchers dealing with biological and technical aspects of neural computation. It appears to be obvious to emphasise modularity in neural systems because the vertebrate brain is highly modular both in an anatomical and in a functional sense.

It is important to stress that there are different concepts of modules. 'When a neuroscientist uses the word "module", s/he is usually referring to the fact that brains are structured, with cells, columns, layers, and regions which divide up the labour of information processing in various ways'

[1]This paper is a revised and extended version of the GECCO 2001 Late-Breaking Paper by Hüsken, Igel, & Toussaint (2001).

(Elman et al. 1996). A definition of modularity from the viewpoint of behavioural or cognitive science could be based on Fodor's work (Fodor 1983): 'A module is a specialised, encapsulated mental organ that has evolved to handle specific information types of particular relevance to the species'. In general, modularity can be viewed as a fundamental design principle of complex systems. For example, the strategy to divide a complicated problem into several smaller subproblems, which may be easier to solve than the whole problem, has led to modular NNs, most often manually designed architectures consisting of *expert networks*, cf. Jordan & Jacobs (1995) and Sharkey (1996, 1997). Already in 1962, Simon (1962) gave very elaborate arguments in favour of modularity. He claimed that the *development* of complex systems requires stable subsystems, which are combined in a hierarchical manner. Similarly, an engineer might argue that it is easier to design a complex system from smaller building blocks that have been tested in other systems and reliably solve a self-contained problem. According to such arguments, modularity could be viewed not only as an outcome of system design, but as a necessity for the development of highly complex systems.

In any case, the modularity of an architecture is a result—or a by-product—of a creation process, e.g. of natural evolution in case of biological systems. In a technical framework, the role of natural evolution is often played by evolutionary algorithms: The task of finding a suitable network for a given problem is often solved by means of evolutionary computation, please refer to the survey of Yao (1999). Based on arguments similar to the ones given above, evolutionary algorithms for the optimisation of NNs have often been designed in such a way that modular architectures are favoured. This can be achieved by indirect encoding schemes, where the adjacency matrix of the graph describing the NN is not directly stored in the genome of an individual, but a non-trivial (e.g. grammar-based) genotype-phenotype-mapping is utilised to construct the NN from a given genotype (Kitano 1990; Gruau 1995; Friedrich & Moraga 1996; Sendhoff 1998; Sendhoff & Kreutz 1999). However, we believe that there are some fundamental, unanswered questions. Above all, we want to know under which circumstances modular architectures of neural systems evolve. This basic question raises several others, for example: For what kind of tasks are modular architectures favourable? What are expressive and general measures for modularity?

In this article, we test the hypothesis whether modular problems are solved better by modular NNs than by non-modular ones. This claim appears natural and is often presumed in the literature. We utilise evolutionary algorithms to search for good NNs tackling artificial modular problems. These algorithms have no explicit bias towards modular architectures; *if modular networks are beneficial, they will evolve.* If no increase in modularity is observed, the hypothesis has to be reconsidered. Measures that quantify modularity are proposed and employed to trace the evolution of modular architectures. Different tasks and both Lamarckian and Darwinian inheritance are considered in order to test the task-dependency of the hypothesis. In particular, we want to verify if the demand for *fast* (gradient-based) *learning* increases the need for modularity.

Our investigation is related to the work of Bullinaria (2001) and Di Ferdinando et al. (2001), who studied the non-Lamarckian evolution of architectural modularity in NNs with a single hidden layer in the context of the "what" and "where" problem (Rueckl, Cave, & Kosslyn 1989). They found that modular architectures develop to improve the learning performance, i.e. the networks evolve such that some of the hidden neurons are connected only to "what" or only to "where" output neurons. We extend these results by introducing finer-grained measures for different aspects of the modularity of arbitrarily connected feed-forward networks (considering the structure and the weights) and by investigating the influence of different constraints (e.g. the scheme of inheritance of the weights or the goal of the optimisation).

In the next section, we define a concept of modularity of NNs and introduce measures for its degree. In section 3, we present our simulations, and in the succeeding section, we discuss the results. The article ends with a conclusion and an outlook.

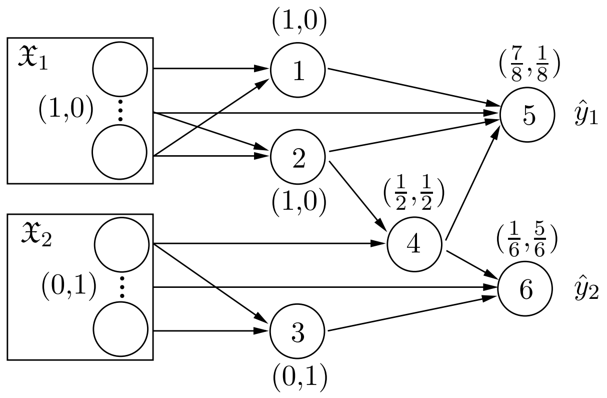

Figure 1: Sample architecture to illustrate the measure of the degree of modularity.

2 Modularity of neural networks

We concentrate on feed-forward NNs. The architecture of a NN can be defined by a directed graph, where the vertexes correspond to the neurons and the edges indicate the flow of information. The performance of a NN depends on both its architecture and the weights that are associated with its edges. Usually, gradient-based optimisation is applied for the adjustment of the weights, whereas the search for a suitable architecture for a given task sets up a hard discrete optimisation problem, which is tackled successfully by means of evolutionary algorithms (Yao 1999). Evolution of connectionist models is not only carried out to improve the performance of NNs for technical purposes, but also to investigate fundamental issues like the interaction between evolution and learning (Nolfi & Parisi 1999).

Most people have an intuitive idea about what modular architectures and networks are. Usually, they refer either to structural modularity, defining modules as highly connected areas that are only sparsely connected to the surrounding, or to functional modularity, see Snoad & Bossomaier (1995) and Toussaint (2002b) for different approaches. However, giving a general mathematical definition and providing a measure for the degree of modularity capturing all concepts of modularity mentioned above and in section 1 is a formidable task. Hence, we do not propose a general definition of modularity, but focus only on a specific character of problems and architectures: We call a problem modular if it can be solved by a number of non-interacting subsystems.

We assume that a NN has to deal with a completely separable problem with m output variables $y_1, \dots, y_m$. Separable means that the set of input variables $x_1, \dots, x_n$ can be divided into m disjoint subsets $\mathcal{X}_j$ such that the underlying input-output mapping

$$(y_1, \dots, y_m) = f(x_1, \dots, x_n) \tag{1}$$

can be expressed as

$$f = (f_1, \dots, f_m) \quad , \quad \text{where } f_j \text{ depends only on variables in } \mathcal{X}_j. \tag{2}$$

This definition is an extension of the one used by Lam & Shin (1998).

Given a separable problem, it does not seem to be disadvantageous to divide the whole network into m separated ones; such a system would be called highly modular. In this context, the degree of modularity of the network is related to how strong separate parts interfere. In the following, we will give two possible definitions of such a measure.

First, the neurons can be distinguished by their source of information presuming that m and the partitioning of the inputs are known: Neurons that solely get input, directly or indirectly, from one subset $\mathcal{X}_j$, $j \in \{1, \dots, m\}$, are denoted as *pure* neurons. In the example in figure 1, neurons

1, 2, and 3 are pure. Neurons 4, 5, and 6 are *mixed* to a certain degree, as they receive input from both subsets of input neurons.

The degree of modularity $\mathcal{M}$ is defined as the average degree of pureness of the hidden and output neurons, calculated by the following procedure: First, an m-tuple $(d_i(1), \ldots, d_i(m))$ is assigned to each neuron i, indicating the degree of dependency of the neuron on the m different input subsets. For the input neurons all $d_i(j)$ are zero except the one with j equal to the index of the subset the input neuron belongs to; this one is set to one:

$$d_i(j) = \begin{cases} 1 & x_i \in \mathcal{X}_j \\ 0 & x_i \notin \mathcal{X}_j \end{cases} . \tag{3}$$

The values $d_i(j)$ of the hidden and output neurons are defined recursively:

$$d'_i(j) = \sum_k |w_{ik}|\, d_k(j) \quad \text{and} \quad d_i(j) = \frac{d'_i(j)}{\sum_{j'=1}^{m} d'_i(j')} \quad . \tag{4}$$

The first sum runs over all neurons k the i-th neuron gets input from and w_{ik} is the weight of the connection from neuron k to neuron i. Neglecting the weights, i.e. assuming all of them to be equal, yields in the above example the 2-tuples depicted in figure 1. In the following, we quantify the degree of pureness of each neuron i by means of the variance $\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}\left(d_i(j) - \frac{1}{m}\right)^2$; the higher σ_i^2, the higher the pureness of neuron i. It can be shown that $\frac{m-1}{m^2}$, the maximum value of σ_i^2, is only reached by pure neurons. The modularity measure of the network is given by the average variance of all N hidden and output neurons

$$\mathcal{M}^{(\text{weight})} = \frac{m^2}{m-1}\frac{1}{N}\sum_i \sigma_i^2 \quad , \tag{5}$$

mapped into the interval [0,1] by means of the first factor. If the weights are neglected in equation (4) (i.e. only the architecture becomes important) we denote the measure defined by equation (5) as $\mathcal{M}^{(\text{arch.})}$. In both cases, a completely separated network corresponds to a value of $\mathcal{M} = 1$, a homogeneous one to $\mathcal{M} = 0$. In case of the example in figure 1, we have $\mathcal{M}^{(\text{arch.})} = 577/864 \approx 0.668$.

As these measures focus on the basic information processing units, they allow for an identification and separation of modules if modules are defined as those subgraphs with input from the same subproblem; this definition is related to the concept of strongly connected regions with a low degree of connectivity to the surrounding. According to our definition, modules are determined in a "bottom-up" fashion, a procedure that appears suitable to quantify to which extend a network can be regarded as a hierarchical assembly of modules.

Usually, there is no strict partitioning of the input space as in our scenario and even if there is such a division, it is unlikely to be known. However, the measures $\mathcal{M}^{(\text{weight})}$ and $\mathcal{M}^{(\text{arch.})}$ can alternatively be calculated in a "top-down" fashion, i.e. starting from the outputs as pure neurons, where the partitioning is usually known, and summing over the *output* connections of the i-th unit in the calculation of (4). The resulting "top-down" version of $\mathcal{M}^{(\text{arch.})}$ can be regarded as a finer-grained version of the modularity measure used by Di Ferdinando et al. (2001).

As an example, we monitor the "top-down" version of $\mathcal{M}^{(\text{weight})}$ during the learning process of a fully connected NN for the not completely separable "what" and "where" problem as described by Rueckl et al. (1989) (except that we use batch learning as described below). Figure 2 shows that first the "where" and then the "what" sub-problem is learnt by the NN (compare dash-dotted line vs. thin dotted line). This is also reflected by the modularity measure: during the initial phase of learning, the "where" problem dominates over the "what" problem (bold solid line) and the overall modularity strongly increases (bold dotted line). While adapting to the "what" problem, a restructuring process can be observed: the focus on the "where" task is reduced while the overall modularity decreases and subsequently—after restructuring—increases again. These results are

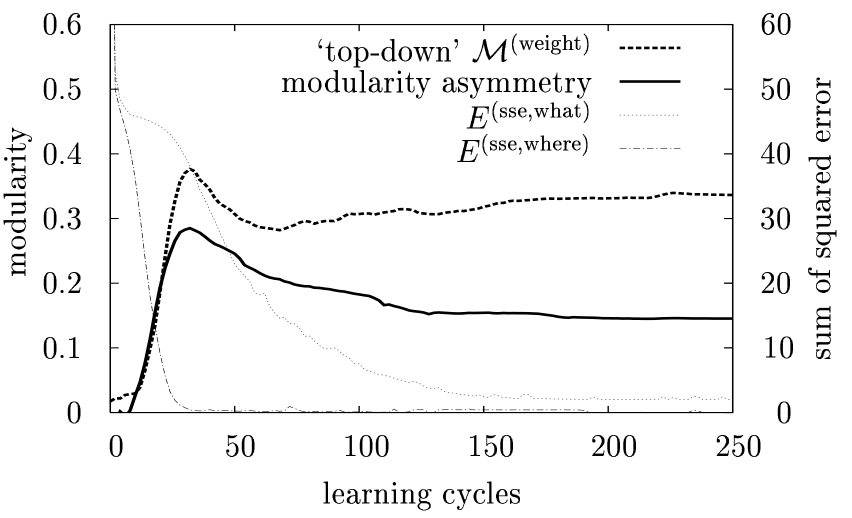

Figure 2: Development of the weight-modularity during learning the "what" and "where" problem with a fully connected architecture with a single hidden layer with 18 neurons. Shown are the medians of 10 independent learning trials. The sum of squares error used for training is shown separated into its "what" and "where" part. The bolt dotted line depicts the "top-down" version of $\mathcal{M}^{(\text{weight})}$ and the solid line characterises the mean asymmetry $\frac{1}{2}(d_{\text{where}} - d_{\text{what}})$ at the neurons, i.e. the larger this value, the stronger the neurons are connected to the "where" problem.

in accordance with the finding of Rueckl et al. that the "where" sub-problem is learnt first and additionally demonstrate that even in a fully connected network the development of modularity during learning can be measured. In the following we focus on the evolution of modularity.

3 Experimental framework

In this section, we present the framework of our simulations of the evolution of NNs. In the experiments, we distinguish between two different kinds of application of the optimised NNs, in the following denoted as two different *tasks*. A NN is denoted *accurate model*, if it is designed for solving a single problem, for instance representing an input-output-mapping induced by a sample data set, as accurately as possible. The search for an accurate model is the most frequent application of NN optimisation. The other task is to be a *fast learner*, an architecture that can learn a given problem within a short time. As this fast learning task is most sensible when dealing with a number of different but related tasks, the problem to be learnt is changed in every generation, but remains separable in the sense of section 2. To make the results of the fast learner more comparable to the accurate modelling task, we also performed fast learner optimisation trials without changing the problem in every generation, i.e. the same problem has to be learnt repeatedly as fast as possible. In the following the superscripts $^{(\neq)}$ or $^{(=)}$ indicate, whether the problem is periodically changed or not. In section 4, we will see that the choice of the task has a strong impact on the development of modularity of the optimisation outcomes.[2]

[2]The distinction between these two kinds of tasks is also discussed by Hüsken et al. (2000); it is shown that the fast learner should be preferred if it is necessary to cope with a class of problems, i.e. to be able to adapt to a number of different, but related problems, in particular within a short time. The modularity may be one of the main differences between the NNs evolved by Hüsken et al. for the different tasks.

3.1 Data set

We choose a completely separable problem in the sense of section 2. We use two different classification problems ($m = 2$) with three Boolean inputs each ($n = 6$). For each evolutionary trial the class labels corresponding to the inputs of each problem are chosen randomly such that half of the input patterns belong to each class. The data set contains all possible combinations of inputs (i.e. $2^n = 64$ different patterns), "true" and "false" are encoded by 1 and -1, respectively.

3.2 Neural networks

We use NNs with six input and two linear output neurons. The absolute value of the measure of modularity is strongly influenced by the number of hidden neurons and connections. In particular, it has turned out that very modular NNs ($\mathcal{M} \approx 1$) evolve in case of a slight selective pressure towards small NNs. To avoid side-effects, we keep the size of the NNs constant. Our experiments are performed with ten hidden neurons with sigmoidal activation function and 40 connections. The resulting networks are large enough to solve the problem, but sparse enough to allow for the evolution of separated modules. However, experiments with slightly modified architecture sizes yield qualitatively similar results. There is no maximum number of layers in the architecture, the only restriction is that each hidden neuron must be connected, directly or indirectly, to at least one input and one output neuron.

Training is performed by means of iRprop$^+$ (Igel & Hüsken 2002), an improved version of the Rprop-algorithm (Riedmiller & Braun 1993), which is a powerful, gradient-based batch-learning algorithm. The aim of training is the minimisation of the mean squared error $E^{(\mathrm{mse})}$ (i.e. the mean squared difference between the NN's outputs $\hat{y}_j$ and the target values y_j). We compute the classification result of the network by mapping positive outputs to "true" and negative ones to "false". The classification error $E^{(\mathrm{class})}$ is given by the rate of incorrect classifications.

3.3 Evolutionary algorithm

Prior to the first generation, $\mu = 10$ individuals are initialised at random. The 40 connections are randomly spread over the whole architecture and the weights are initialised with values from the interval $[-0.2, 0.2]$. The only operator employed to generate the $\lambda = 70$ offspring per generation is the random move of single connections (i.e. a connection is deleted and afterwards inserted at a random position and reinitialised randomly). This means that the number of connections remains constant, see above. The parents of the next generation are selected out of the offspring by means of (μ,λ)-selection, i.e. the best μ out of the λ offspring form the next parent population (Schwefel 1995).

Depending on the task of the optimisation (accurate model or fast learner) the algorithms differ slightly. In the former case, in every generation each individual is trained for $T^{(\max)} = 200$ cycles. In the *Lamarckian* trials, learning starts from the weights that have been reencoded after learning in the previous generation. In the *Darwinian* style evolution, only the architecture is encoded in the individual's genome and the weights are reinitialised prior to learning within the interval $[-0.2, 0.2]$. The fitness function

$$\phi^{(\text{accurate model})} = -\left(E^{(\mathrm{class})} + \alpha_2 E^{(\mathrm{mse})}\right) \tag{6}$$

includes only the errors after learning.

In case of the evolution of the fast learner, in every generation the weights of each individual are reinitialised randomly in the interval $[-0.2, 0.2]$ (Darwinian style) and learning takes place as long as there are still misclassified patterns left, but not for more than a maximum number of

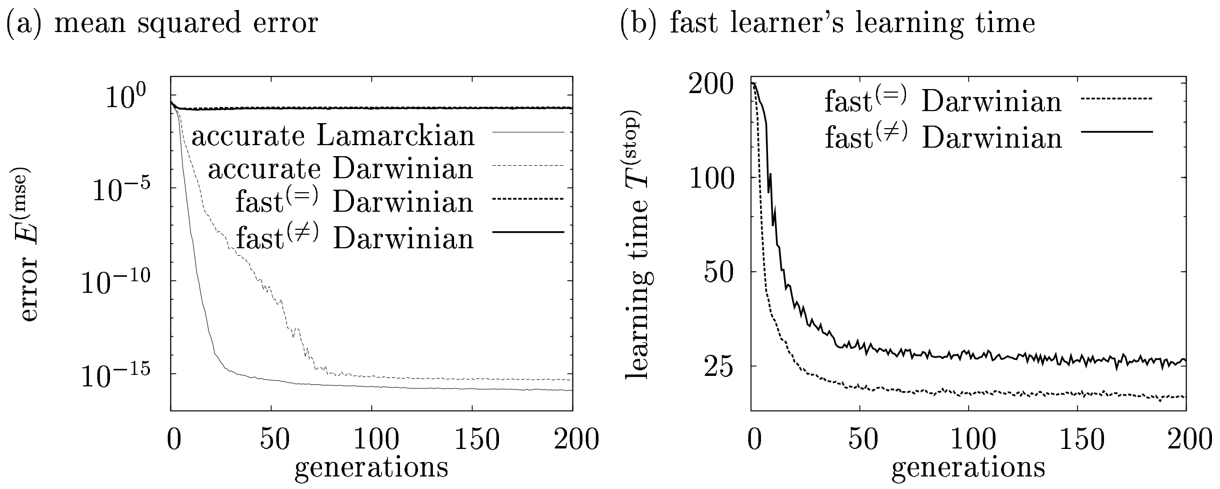

Figure 3: Relevant terms of the fitness function.

$T^{(\max)} = 200$ cycles. In this case, the fitness is given by

$$\phi^{(\text{fast learner})} = -\left(E^{(\text{class})} + \alpha_1 \frac{T^{(\text{stop})}}{T^{(\max)}} + \alpha_2 E^{(\text{mse})}\right) \quad . \tag{7}$$

Herein, $T^{(\text{stop})}$ denotes the cycle when learning actually has stopped. The choice of $\alpha_1 = 10^{-4}$ and $\alpha_2 = 10^{-8}$ effects that $E^{(\text{class})}$ has the strongest and $E^{(\text{mse})}$ the weakest influence on the fitness, i.e. the primary aim of the optimisation is the correct classification, which cannot be levelled out by faster learning or a smaller $E^{(\text{mse})}$.

4 Discussion of experimental results

All graphs represent the median of the results of 200 independent trials. Figure 3 depicts the evolution of the relevant fitness terms $E^{(\text{mse})}$ and $T^{(\text{stop})}$. The classification error is hardly of interest, because it vanishes already in generation 6 for all four models. In the succeeding approximately 100 generations, the algorithms focus on the reduction of the learning time and the mean squared error, respectively, and thereafter, no evolutionary progress is observable.

Figure 4 displays the development of the "bottom-up" *architecture-modularity* $\mathcal{M}^{(\text{arch.})}$ and *weight-modularity* $\mathcal{M}^{(\text{weight})}$ for all four experimental scenarios. We find that except for some random fluctuations the modularity increases, the more the fitness decreases. This general increase of modularity supports the intuition that modular networks are indeed advantageous for modular problems. The following observations, to be discussed below, give a more detailed insight into the relation between the two modularity measures and the four experimental scenarios:

1. Figure 3 (a) clearly exhibits that the two accurate learners (accurate Darwinian and accurate Lamarckian) efficiently decrease the $E^{(\text{mse})}$, while the fast learners (fast Darwinian) minimise the classification learning time $T^{(\text{stop})}$, Figure 3 (b), without reaching small $E^{(\text{mse})}$.

2. Figure 4 shows that the fast learners reach significantly higher architecture-modularity $\mathcal{M}^{(\text{arch.})}$ than the accurate learners (Wilcoxon rank sum test, $p < 0.05$ after generation 50); whereas the accurate learners reach significantly higher weight-modularity $\mathcal{M}^{(\text{weight})}$ than the fast learners ($p < 0.05$ after generation 10).

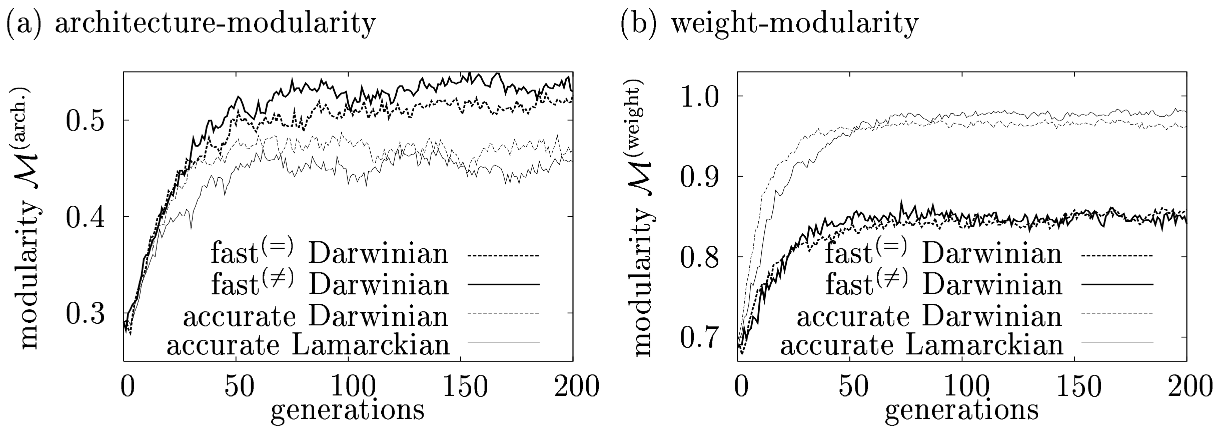

Figure 4: Average modularity in the parent population.

3. Comparing the two fast learners, we see that the reiterated change of the problem yields a higher architecture-modularity $\mathcal{M}^{(\text{arch.})}$ ($p < 0.05$ in most generations), while the weight-modularity remains unchanged.

To understand observations 1 and 2, let us first reconsider the relation between $\mathcal{M}^{(\text{arch.})}$ and $\mathcal{M}^{(\text{weight})}$. It is plausible that a high architecture-modularity $\mathcal{M}^{(\text{arch.})}$ also induces a high weight-modularity $\mathcal{M}^{(\text{weight})}$, since architecture determines the terms of summation in (4). In particular, $\mathcal{M}^{(\text{arch.})} = 1$ implies $\mathcal{M}^{(\text{weight})} = 1$. However, the inverse need not be true; even a network with a non-modular architecture may reach high $\mathcal{M}^{(\text{weight})}$ by having many weights of "cross-connections" set to zero. In this sense, a suitable adjustment of weights can counterbalance a non-modular architecture in favour of weight-modularity $\mathcal{M}^{(\text{weight})}$.[3]

A fine tuning of the weights should lead to a small $E^{(\text{mse})}$. This explains what we find in figure 4 (b): both algorithms that minimise the $E^{(\text{mse})}$, the accurate Darwinian and accurate Lamarckian, have high weight-modularity, even close to 1. In contrast, the fast Darwinian learners do not have a high weight-modularity (the $E^{(\text{mse})}$ is orders of magnitude higher) but instead reach high architecture-modularity, see figure 4 (a). The task of learning fast does not aim at a precise adjustment of weights, it aims at an optimal architectural predisposition to learn the classification. The learning is stopped after reaching this goal, so that a fine adjustment of weights cannot take place. Hence, the fast learners rely on appropriate architecture-modularity, whereas the accurate models can counterbalance architectural imperfection by a more accurate weight adjustment in the longer learning period. Summarising, the evolutionary pressure towards architecture-modularity is increased by the task of fast learning. In turn, the reduction of the $E^{(\text{mse})}$ for modular problems increases weight-modularity also when the architecture is not particularly modular, and thus induces lower pressure towards architecture-modularity. The latter effect depends only weakly on the scheme of inheritance.

Regarding observation 3 the architecture-modularity $\mathcal{M}^{(\text{arch.})}$ is maximal when the networks have to learn a number of different, but related (in the sense of section 2 and section 3.1) problems. If the same problem is to be learnt repeatedly, certain "cross-connections" do not necessarily interfere with learning, depending on particular characteristics of the problem and of the learning algorithm. However, if the problem is changed over and over, at least for some of the problems these "undesired" connections will interfere with learning. Therefore, it becomes more important to adapt the architecture to the common aspects of all presented problems; in our example this

[3]It is likely that an efficient learning algorithm will adjust the weight of an undesirable connection to a small absolute value. Especially, the fact that we use a *batch* learning method leads to the speculation that the problem's modularity is "detected" very quickly by the learning algorithm since the gradient calculation averages over all possible cases. Thereby all correlations between the subproblems are cancelled out.

corresponds to an increase of $\mathcal{M}^{(\text{arch.})}$. The need for fast and robust learning demands for more modular architectures.

In addition, we performed the same experiments under changed conditions: We considered sigmoidal output neurons instead of linear ones and applied (i) the $E^{(\text{mse})}$ cost function with target values 0.1 and 0.9, (ii) the $E^{(\text{mse})}$ cost function with target values 0 and 1, and (iii) the Cross-Entropy cost function (with target values 0 and 1). In the first case the results are qualitatively the same as the ones presented here. However, in the second and third case the results cannot be reproduced. The reason may be the unbounded learning dynamics towards saturated output neurons in the second and third case, where the absolute weight values are orders of magnitude greater than in the other scenarios. This effect could also explain the results found by Bullinaria (2001) that the use of a $E^{(\text{mse})}$ cost function with target values 0.1 and 0.9 lead to different degrees of modularity than the Cross-Entropy cost function.

5 Conclusion and outlook

The presented work aims at a better understanding of the development and functional importance of modularity in neural networks (NNs). We introduced measures for the degree of modularity in NNs that allow to characterise the architecture and the weight configuration. The measures proved to be of use to understand the development of modularity during learning and evolution of NNs.

We designed our experiments in order to distinguish between different optimality criteria (learning a classification maximally fast or learning a regression maximally accurate in a given time) and different inheritance schemes (Lamarckian, with inheritance of trained weights, and Darwinian, without such weight inheritance). Our results show that modularity increases with the NN's efficiency, but that this increase is task-dependent: The fast learning criterion without weight inheritance (significantly) enforces the development of architecture-modularity; the accurate regression criterion (with and without weight inheritance) instead enforces the development of a modular weight configuration. Generally, we gave support to the hypotheses that modular NNs are indeed beneficial for solving modular problems and that the need for modularity becomes the stronger, the more fast and robust learning becomes part of the task definition.

From a broader perspective, our efforts aim at the evolution of large systems with modular information processing. In future work, additional measures of modularity should be developed that address more specific, e.g. information theoretic aspects of modular information processing. Such measures would guide the development of new types of NNs that allow more modularity *a priori* than today's common NNs. If findings confirm the appropriateness of modular architectures for classes of problems, then evolutionary algorithms should be specifically designed to find modular architectures (Toussaint 2002a). New mutation operators or other types of encoding (e.g. recursive or grammar encodings Kitano 1990; Gruau 1995; Friedrich & Moraga 1996; Sendhoff 1998; Sendhoff & Kreutz 1999) should be analysed and (further) developed.

Acknowledgement

We would like to thank W. von Seelen for many beneficial discussions and the anonymous reviewers for their helpful comments and suggestions. Moreover we acknowledge financial support from the BMBF, grant LOKI 01 IB 001 C.

References

Bullinaria, J. A. (2001). Simulating the evolution of modular neural systems. In J. D. Moore & K. Stenning (Eds.), *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, Mahwah, NJ, pp. 146–151. Lawrence Erlbaum Associates.

Di Ferdinando, A., R. Calabretta, & D. Parisi (2001). Evolving modular architectures for neural networks. In R. French & J. Sougne (Eds.), *Proceedings of the sixth Neural Computation and Psychology Workshop: Evolution, Learning and Development*, pp. 253–262. Springer-Verlag.

Elman, J. L., E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, & K. Plunkett (1996). *Rethinking Innateness – A Connectionist Perspective on Development.* MIT Press.

Fodor, J. A. (1983). *The Modularity of Mind: An Essay on Faculty Psychology.* The MIT Press.

Friedrich, C. M. & C. Moraga (1996). An evolutionary method to find good building-blocks for architectures of artificial neural networks. In *Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU '96)*, pp. 951–956.

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior 3*(2), 151–183.

Hüsken, M., J. E. Gayko, & B. A. Sendhoff (2000). Optimization for problem classes – Neural networks that learn to learn. In X. Yao & D. B. Fogel (Eds.), *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000)*, pp. 98–109. IEEE Press.

Hüsken, M., C. Igel, & M. Toussaint (2001). Task-dependent evolution of modularity in neural networks – a quantitative case study. In E. D. Goodman (Ed.), *Late-Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 187–193.

Igel, C. & M. Hüsken (2002). Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*. In press.

Jordan, M. I. & A. Jacobs (1995). Modular and hierarchical learning systems. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 579–582. MIT Press.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems* **4**, 461–476.

Lam, C.-H. & F. G. Shin (1998). Formation and dynamics of modules in a dual-tasking multilayer feed-forward neural network. *Physical Review E 58*(3), 3673–3677.

Nolfi, S. & D. Parisi (1999). Learning and evolution. *Autonomous Robots 7*(1), 89–113.

Riedmiller, M. & H. Braun (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pp. 586–591. IEEE Press.

Rueckl, J. G., K. R. Cave, & S. M. Kosslyn (1989). Why are 'what' and 'where' processed by separate cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience* **1**, 171–186.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. John Wiley & Sons.

Sendhoff, B. A. (1998). *Evolution of Structures – Optimization of Artificial Neural Structures for Information Processing.* Aachen, Germany: Shaker Verlag.

Sendhoff, B. A. & M. Kreutz (1999). Variable encoding of modular neural networks for time series prediction. In *Congress on Evolutionary Computation (CEC '99)*, Volume 1, pp. 259–266. IEEE Press.

Sharkey, A. (1996). On combining artificial neural networks. *Connection Science 8*(3), 299–314.

Sharkey, A. (1997). Modularity, combining and artificial neural nets. *Connection Science 9*(1), 3–10.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society* **106**, 467–482.

Snoad, N. & T. Bossomaier (1995). MONSTER – the ghost in the connection machine: Modularity of neural systems in theoretical evolutionary research. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*. ACM Press and IEEE Press.

Toussaint, M. (2002a). A neural model for multi-expert architectures. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002)*, pp. 2755–2760. IEEE Press.

Toussaint, M. (2002b). On model selection and the disability of neural networks to decompose tasks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002)*, pp. 245–250. IEEE Press.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE 87*(9), 1423–1447.