INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15 – 18, 2005

SPECIFICATION OF INDIVIDUALIZED PRODUCTS BY NEURAL NETWORKS

Maik Maurer, Wieland Biedermann

Keywords: Neural networks, product configuration, product structure, product spectrum

1 Introduction

An increasing number of manufacturers offers customizable product spectra instead of common invariant products. This can be explained e.g. by growing market competition and a steady trend towards individualization in society. Product spectra allow the customization of products to meet customers' requirements by adapting specific attributes, the so-called degrees of freedom. Different methods have already been developed to set up a product spectrum [1], but so far, no efficient proceeding is available to determine a customized product from a spectrum.

One basic assumption of common methods for product specification is the availability of products in discrete specifications [2], e.g. building blocks or type series. These methods focus on the selection of one product variant amongst predetermined possibilities. However, customizable products have to be adapted to customers' requirements in contrast to selecting between given alternatives. For this reason, established procedures do not meet actual demands. In this contribution, we present a neural network approach for specification of customized products. These networks are highly adaptable and represent powerful tools for modeling complex interdependencies, which are often hard to quantify.

2 Basic approach for neural network application

Neural networks are applied to various research areas as well as industrial production with similar approaches for their design. Often recurring, congenerous tasks, as for example pattern recognition or time series prediction are available as discrete program packets and operate with standardized design methods. The here presented approach especially is applied to product specification demands and bases on application of feed-forward neural networks. For their creation the following steps are generally performed: Firstly, the problem description has to be carefully developed, the acquisition of training data follows next. Based on this, information on an adequate network structure must be identified; finally, the network has to be trained in order to permit application to the underlying problem [3].

For applying neural networks, at least required input data, expected output data and the basic sort of problem have to be known. Additional information, e.g. dependencies between input and output data are helpful. Firstly, expected output data of the neural network must be identified, which bases on (often implicitly available) user knowledge. This data mostly contains few parameters, usually independent from each other. When dealing with complex problems, the application of a systematic acquisition approach is recommended and attention

must be turned to probably existing dependencies between output data. It is possible that these dependencies already specify the applicable neural network type. Input data, which has to be subsequently acquired, normally is not of distinct character, as defining dependencies are unknown and to be learnt by the neural network. To acquire input data creativity methods or intuitive collection are used. A more systematic method is to identify system parameters, which can be identified easily and precisely. Mostly, they correspond to meaningful input data.

The collection of training data required for the learning process can also be grouped into collection of product specifying input data and expected output data – both depending on the problem in question. Different methods can be used, e.g. direct measurement or questionnaires. The more input and output data is required for product specification, the more training data sets have to be collected. In-between the value range of input data the training data should cover as many different combinations as possible.

Before defining a network structure (based on case specific information), available data has to be reviewed concerning constant input or output values, which are identical for every data set. Those as well as redundant input can be deleted. The format of input and output data is important to determine a suitable network structure, as it primarily affects the quantity and type of neurons. Data available in continuous values can directly be applied or requires only its adequate scaling. One single neuron is sufficient to represent such data. If input or output only exists in form of discrete and non-numeric values, appropriate coding has to be carried out. At best only two values, e.g. “metal” and “plastic” occur. Here a binary coding is sufficient for representation, i.e. one single neuron realizes the reproduction. However, to deal with three or even more values, every value demands its own neuron for being represented. Each value is coded in binary form, but only one of them can be set to 1. The parameters format and quantity of data sets specify the complexity of hidden neuron layers. Different heuristics exist for their detailed determination [3, 4].

The training of neural networks does not only consist of choice and application of learning methods, but contains all measurements contributing to the network design when problem type and network structure are already fixed. These measures comprehend in either case: choice of an adequate learning method, identification of required parameters for the learning method, and the application of chosen learning method. Further measurements can be: adjustment of decisive learning parameters, adjustment of the network structure, repartitioning of control and training data, application of further training data or optimization of the network. The standard learning method for feed-forward neural networks is backpropagation, which possesses only one learning parameter in its basic occurrence. If a constant learning factor is used, its correct specification is decisive for reaching a minimum failure without extensive learning efforts.

For the implementation of a learning approach, available data sets have to be partitioned in control and training data. Training data sets are the basis for neural network learning; the results must be verified by use of control data. This partitioning can be done regarding different criteria. Random partition is useful to create a network suitable for wide areas of input data. At the beginning of the learning process free network parameters are initialized. This can be done at random (in-between a specific value range) or by use of heuristics, e.g. the rule of Hebb [4]. When applying the learning method, an adequate learning duration is decisive for the resulting quality. Too short learning will probably lead to an undesired large failure minimum. However, too long learning can worsen the generalization aptitude.

3 Application of neural networks to product specification

Based on fundamental knowledge of neural network design an approach is presented, which allows deriving customer specific product definitions from a comprehensive product spectrum. This approach exceeds the common selection between pre-designed product variants. In fact, it permits generating a product specification from an area of realizable possibilities by use of customer requirements, which do not correspond to technical attributes. Particularly product specifications can be derived from continuous attribute areas of element properties (degrees of freedom) permitting more customer orientation than configuration of discrete variant models. The specification of individualized ball pens is chosen as exemplary case study to explain the presented approach with realistic degrees of freedom. For example ball pens can be specified due to color, form and mechanism. To move back the ink store in order to avoid unintended writing, mechanisms based on turning, sliding or pushing exist. Common materials are different sorts of plastic and metal, some components are even made of rubber or wood. The case study is kept at a low level of complexity in order to guarantee comprehension and transferability.

3.1 Product model design

To apply the problem formulation (as the first step of neural network design) to the exemplary case study, a product model has been defined. It is established on an abstract level in order to be applicable to large and varying product areas. Based on this model a complete set of degrees of freedom is worked out, i.e. all product properties have to be acquired defining or affecting the degrees of freedom. By using this model, consistent availability of degrees of freedom for later implementation of the neural network is guaranteed. The result of the approach is a valid product definition, which can be individually produced for a customer. Therefore, the product model is oriented towards production, assembly and customers view to the product (not towards planning and design). The product model, which is shown in figure 1, is composed of three main areas, which are basic elements, properties, and dependencies [5].

A part is the fundamental component of the product model. A part in terms of the product model does not obligatorily consist of exactly one element. It is mainly defined by its function and position in the product. Thus, different forms and sizes can occur. One specification of a part's attributes is called a variant. An assembly represents a collection of parts and possibly further subsidiary assemblies. Finally, a product consists of parts and assemblies. These basic elements are described by properties consisting of attributes and belonging specifications [6]. A specification of an attribute possesses a scale constituting the kind of permitted values. In the presented product model only properties with ratio and nominal scales are considered.

The nominal scale does not possess any ranking and values do not correspond to numbers. An example is the ball pen color, which possesses the specifications blue, red, and green. Ratio scales are number scales with a natural zero point, e.g. lengths.

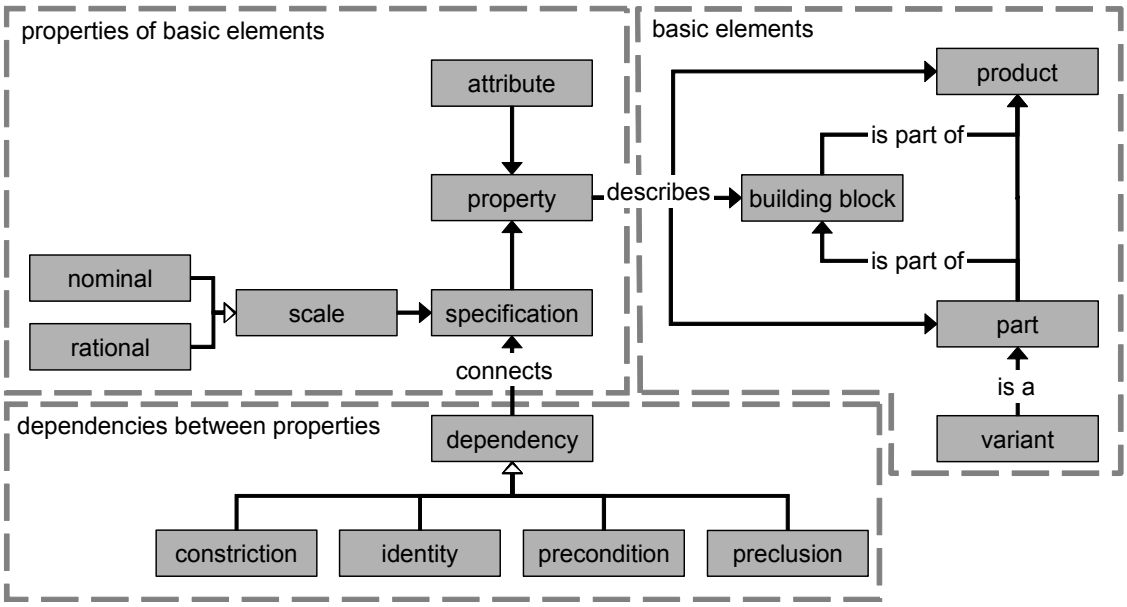

Figure 1. Product model as basis for problem description

The third relevant area of the product model represents the dependencies. These indicate coherences between property specifications, whereas multiple dependencies can appear between two properties. The product model considers four different dependency types: constriction, identity, precondition, and preclusion. The constriction dependency is valid only when relating rational properties. A similar dependency could be created between nominal properties, but can be ascribed to several exclusion dependencies. The constriction dependency diminishes the value range of a second property depending on the actual value of the first one. E.g. this dependency exists between ball pen tube and ink store; the smaller the inner diameter of the tube, the smaller the maximum diameter of the ink store. The identity dependency exists between nominal as well as rational properties. However, associated properties must possess the same scaling. The precondition and preclusion dependencies are relevant only between nominal properties. Precondition means that one property needs to adopt a specific value in order to allow a second one to obtain a desired value. Preclusion dependencies describe that one property does not dare to adopt a specification because of the simultaneous appearance of another property's specification. Constriction and precondition dependencies are directed, i.e. they point from the first property at the second one. However, identity and preclusion dependencies can be reversed, i.e. they are undirected. Dependencies can refer to parts, assemblies, or even the entire product. By combination of here presented dependencies further and more complex dependencies can be defined, e.g. master properties describing comprehensive assemblies.

3.2 Problem description

Based on the product model the determination of problem formulation is considered closer referring to creation of customer individual products. A product spectrum with principal solutions as well as fundamental property description is the initial point for defining a customized product. Thus, the process of specification of all required properties has to be described. The product properties, the scales of each property, and dependencies between the properties have to be known in order to enable product's definition. A schema of required process steps is shown in figure 2.

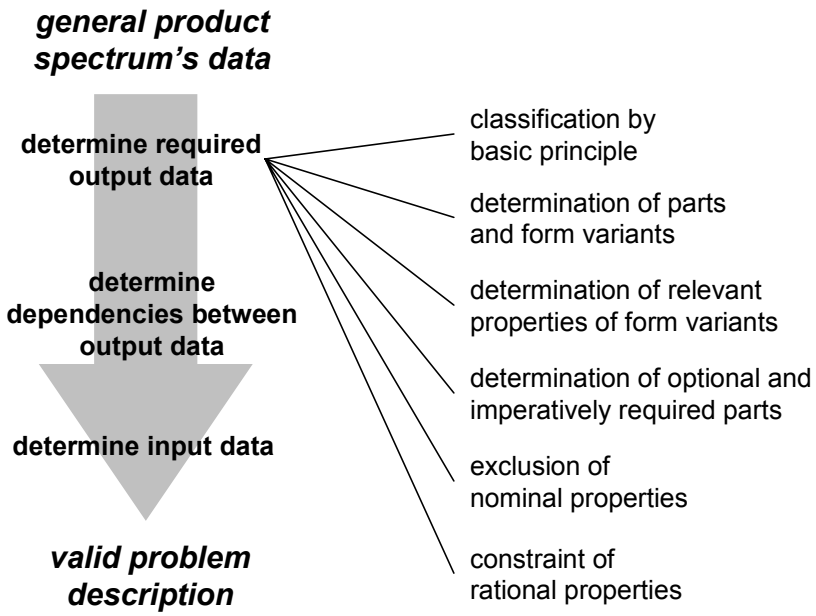

Figure 2. Process for designing the problem description

First, output data is to be determined, which permits the complete product description. I.e. specifications of all properties of the final product are to be derived from resulting output data. For this step dependencies between properties are of major importance, as single property specifications will often determine further ones. Thus, not all specifications have to be determined explicitly in most cases. A subdivision of the product by its basic functions is useful to identify the dependencies. The product is grouped by models, which differ in principal functionality and therefore are designed differently. If this method does not match, often a subdivision by an important auxiliary function can be processed. Thus, the main function of a ball pen is "ink painting". As this function is realized for all ball pens in the same manner, the main function cannot be used for functional subdivision. However, an important auxiliary function is "avoid unintended writing", which can be realized by three different principles. The ink store can be integrated in a non-detachable way in the ball pen and be covered by a cap (principle is called fixed ink store). The other two principles allow a movement of the ink store in order to pull its end back into the ball pen tube. This can be done by a press button combined with a spring (press button mechanic) or by turning a screw guidance (screw mechanic). All three principles differ widely in basic design, assembly, and consisting parts. By subdividing into three models, a first important property of the exemplarily presented product is identified.

Furthermore, all product parts (and applied form variants) have to be acquired. Useful strategies are e.g. designer's enquiry or analysis of design documents. The mentioned second strategy is usually less extensive, especially when treating complex products. When carried out systematically a complete part collection results. Design documents have to be structured (best supported by use of existing PDM-systems) resulting in groups, which contain at least one document for each group. If several documents (describing parts) are assigned to the same group, it must be determined for every product specification, which one can be applied to the specific case. For further investigations, it is useful to identify those documents that are derived from other ones. This leads to a network of derived elements [7], as one form variant can take elements of several other ones.

When the basic functional principles, parts, and form variants are identified, for each form variant the most important properties are to be determined. In particular, these are geometric

properties, e.g. lengths or diameters. This step can be rather demanding, depending on quantity of form variants. If standardized parts appear in the product (e.g. ball pens often use the ink store models ISO 12757), this information can be used in order to reduce the effort. Two important maxims for identifying form variant properties are that object properties do not need to describe the parts completely and property's abstraction level must be chosen due to the specific case. Thus, the most important properties of form variants need to be specified in their geometry only approximately. In consequence, relevant properties are main and connection dimensions. Object properties are inheritable to variant forms derived from them. If a network of derived elements has been designed as mentioned above, the effort of determining derived parts can be reduced by inheriting such properties. In addition to geometric properties, materials of variant forms have to be considered as further relevant properties.

If all product properties are available, their quantity has to be reduced in order to receive required output data of the neural network. Nominal product properties are mainly the existence and form variants of a part as well as its material. In best case properties can only adopt one specification. Such constant properties are not to be determined by the neural network. If more than one specification can be adopted, dependencies between properties have to be considered closer. For investigations on existence and form variant properties of a part, the product assembly supports identifying the required dependencies. One starts with the fixed product parts. As their existence property possesses categorically the specification yes, it is no longer to be considered. An optional part with possible variation in its specification of the existence property can only be applied to the product, if the product possesses an adequate interface. This must be delivered either by a form variant of a fixed part or by another optional part. If a form variant of a fixed part is required by existence of an optional part, existence of the optional part is sufficient as network output to determine both properties. However, if the existence of an optional part is required for integrating another optional part, the existence of the second one will not necessarily be sufficient for determining the first one. This can be clarified at the ball pen example. For its product integration, the optional clip requires the optional cap (as the clip is fixed to it). However, the cap can also be applied to the product without the clip (in another form variant). Furthermore, it is possible that existence of optional parts is not required as output information, when another optional part requires its existence. If a part is categorically required by an optional part and does never exist in the product for itself, both parts should be merged (the object properties have to be newly identified). However, if the part is applied to the product due to several optional parts (which are independent from each other), it has to be kept as stand-alone part. The property type "material of form variant" can be treated in the same manner as it is done with property types explained so far. Dependencies help to exclude further material properties, especially the precondition dependency.

For constriction of rational properties the main dimensions, diameters, and lengths have to be considered. The constriction and identity dependencies are to be examined for rational properties in the product model. If properties are interrelated by the identity dependency, only one of both specifications is to be determined by neural network application. The identity dependency can easily be determined by knowledge of interfaces, wherein identities mostly appear. The constriction dependency is not applicable to reduce properties, as it only constricts the value range of possible property specifications and does not supersede their determination.

If the properties are determined and reduced, the required output data is available. Between outputs further logic dependencies may exist, which can be used for network structure definition. For example dimensions and form variants of an optional part, which does not

exist in an actually concerned product specification, do not have to be determined. If network outputs interrelated by constriction dependencies, network learning is more difficult, because transfer from input to output data as well as dependencies between the output data must be learnt. A possibility to facilitate the learning problem is to constrict the value range of concerned outputs independently from other outputs, so that constraint dependencies do not match any more. This can supersede some outputs, as they are already specified. However, if single outputs would not possess a valid specification, this possibility of facilitation cannot be applied.

In general, input data is to be determined in the same manner as output data. Inputs must completely describe the product from customer view. Always-available parameters can be used as available measurements, but only parameters are reasonable, which can be quantified by customer interaction or can be obtained by customer enquiry. In the ball pen example e.g. the measurement of customer's hand or a web interface enquire would meet these requirements. The abstract customer requirements for the ball pen were acquired in a team-based brainstorming: child proof, non-skid, leak proof, non-allergic, pocked sized, lightweight, not too lightweight, printable, elegant, robust, archivally safe. The parameters have to be expressed numerically in order to be applicable to neural networks. Thus, each parameter specification is assigned to an integer value between 1 and 10. This scaling represents property's importance to the customer (1 corresponds to lowest importance, 10 to highest importance).

3.3 Acquisition of training data

Training data must be collected and possible inputs constricted in order to enable a learning aptitude of the neural network. Training data can be e.g. obtained by customer interviews in form of standardized enquiries. An indirect possibility would be to take notes of sales conversations, but this implies possible interpretations. These can influence training data quality either in positive or negative way, depending on vendor's capability to assess the expressed statements. For acquisition of training data in the ball pen example an enquiry based on MS-Excel has been designed and is displayed in figure 3.

The customer declares his personal weighting of default requirements as well as his individual requirements in the upper part of the user form. In the lower part, he configures the ball pen to fit to his personal liking (from a technical point of view). The product logic of possible combinations and exclusions is implemented in the user form. Thus, only valid product configurations can be determined. Employees and students of the Technical University of Munich executed the enquiry and generated about 100 data sets for further operation.

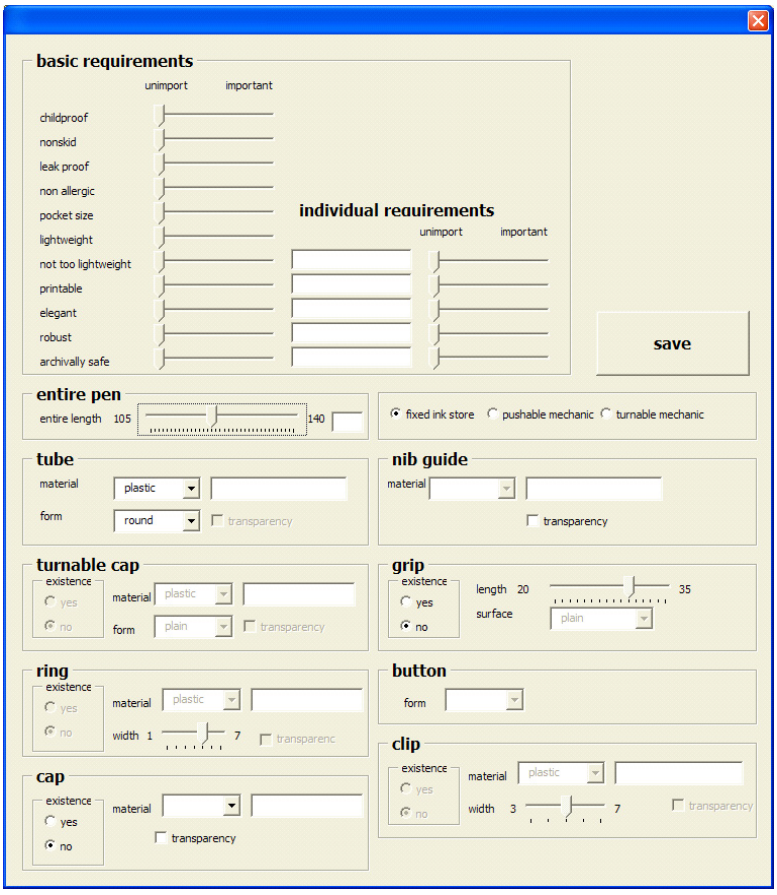

Figure 3. Training data acquisition for the product spectrum of a ball pen

3.4 Neural network design and implementation

When creating a neural network, the structure, which contains the neurons and their dependencies, is to be set up first. The rough structure is already determined by input and output data. Based on the dependencies between them it is possible to subdivide the outputs and to compute them by smaller (and less complex) neural networks. For the ball pen example the subdivision can be carried out by the basic design principle. As every principle possesses its own part structure, it also requires own outputs. The exemplary network division is shown in figure 4.

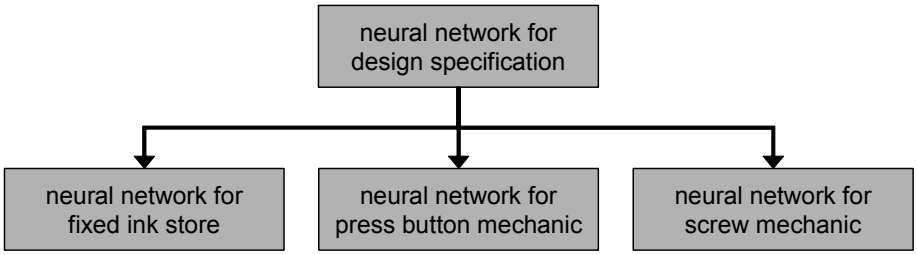

Figure 4. Subdivision of exemplary neural network by basic design principle

The further proceeding is explained with the sub-network designed for the fixed ink store principle. Table 1 displays the required outputs for specifying a ball pen with fixed ink store. The outputs are already referred to their belonging parts. In-between these outputs different dependencies exist, e.g. the length and the surface of the grip only are to be determined if their specified material is set to plastic. The level of detail for subdividing the product design into sub networks depends on the available training data quantity as well as the experience of designers. In the example the network for defining a fixed ink store specification is divided in three sub networks; one for the specification of general details, another for the clip design and the last one for specifying the grip (see figure 4).

Table 1. Parts with required outputs for fixed ink store design

part	output	possible values
entire ball pen	length	105 – 140 mm
tube	form material transparency	Round or hexagonal metal or plastic yes or no
nip guide	material transparency	metal or plastic yes or no
cap	material transparency	metal or plastic yes or no
clip	existence material transparency width	yes or no metal or plastic yes or no 3 – 7 mm
grip	existence length surface	yes or no 25 – 40 mm wave-shaped, plain, knobbed

The network design is displayed for the sub-network describing the grip. Firstly, input and output data is to be formatted according to neural network requirements. The input data must be available as integer numbers or has to be coded adequately. The output data has to orientate on neuron's output functions in order to be computed by the neural network. Typical output functions are identity and sigmoide [Patterson]. Identity can adopt arbitrary real values. Thus, it can be used for representing rational output data. Sigmoide is the standard output of neural networks, which are planned to be trained by backpropagation. Sigmoide can adopt values between 0 and 1. Each property specification must be represented by a single neuron in order to display nominal properties by sigmoide output neurons. If only two specifications occur, one neuron is sufficient. The inputs for determination of the grip are binary-coded; input weighting from 1 to 5 was substituted by 0, weighting from 6 to 10 by 1. The three possible form specifications of the grip were represented by binary values as well. The resulting data set was converted and transferred to the simulation system JavaNNS [8, 9].

Now, the inputs and outputs as well as the data format are defined by coding and transferring. Based on this, the input and output neurons can be defined. The quantity of neurons in the hidden layer is to be estimated to complete the fundamental network structure. In the ball pen example, seven neurons were chosen in the hidden layer, according to formulas from [3]. Thus, the structure is set up with eleven input neurons, seven hidden neurons and four output neurons, as it is displayed in figure 5.

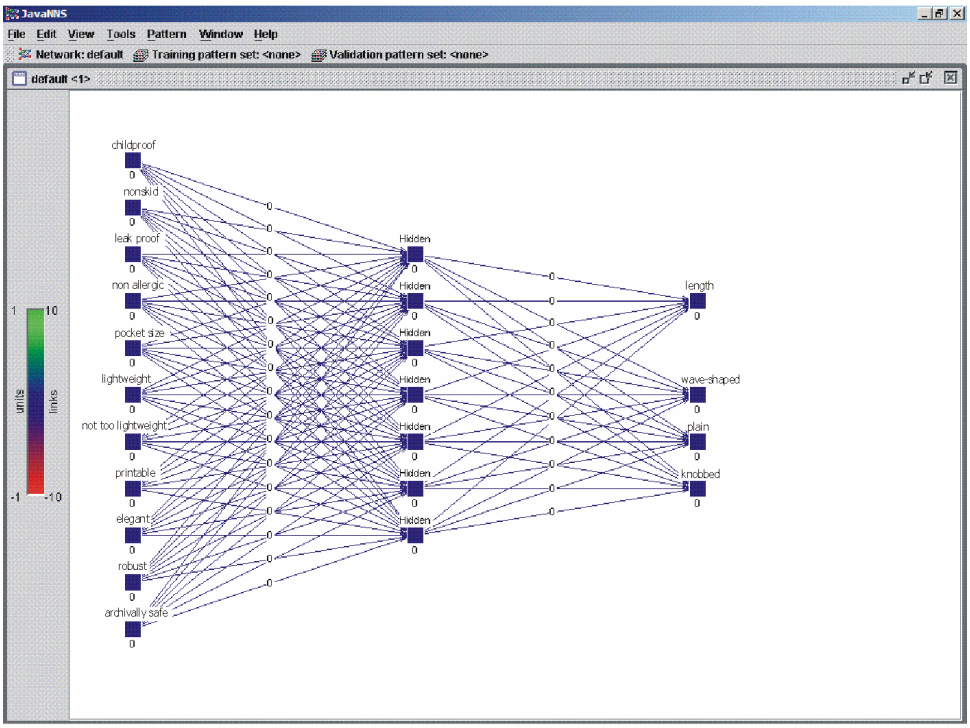

Figure 5. Basic neural network design for grip determination

In the next step, an adequate learning procedure is to be chosen; as the basic network type is a feed-forward network, the backpropagation procedure is obvious. Figure 6 displays the resulting trend of mean squared error referred to the learning cycles. The desired failure value is 0.1 [as recommended by 10], which is under-run after 300 cycles for the first time. After 350 cycles, this failure value is no longer exceeded. The fact that the desired failure value is obtained relatively fast indicates a too large quantity of hidden neurons [4]. This leads at one side to a high quality of adaptation, at the other side the generalization decreases. For this reason, the quantity of hidden neurons was progressively reduced, until the failure trend changed noticeable with three hidden neurons. Now, the failure value of 0.25 is no more under-run, as it can be seen in figure 7.

Two possibilities exist for further attempts of optimization; either the quantity of hidden neurons is increased again or the learning procedure is changed [3]. Thus, the backpropagation with impulse term [4, 10] is chosen and leads to a constant failure value of 0 after 300 learning cycles. Again, the low failure value indicates the possibility of further reduction of hidden neurons. This result is a failure trend with a minimum failure of 0.25 by use of only two neurons in the hidden layer. A further chance of the learning procedure to Quickprop [3, 4] results in a minimum failure of 0 again, as can be seen in figure 8. It does not make sense to further reduce the hidden neurons, as only one neuron would remain. However, one neuron only allows representing a binary decision, while four values are to be determined in the example. Thus, the finally obtained neural network structure with two hidden neurons is shown in figure 9.

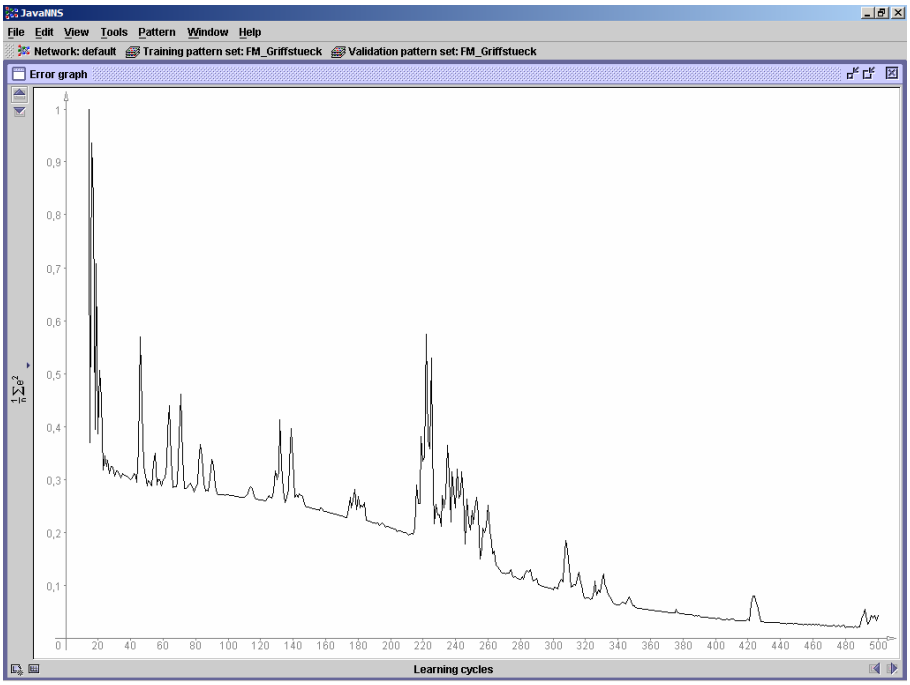

Figure 6. Failure trend for backpropagation learning with seven hidden neurons

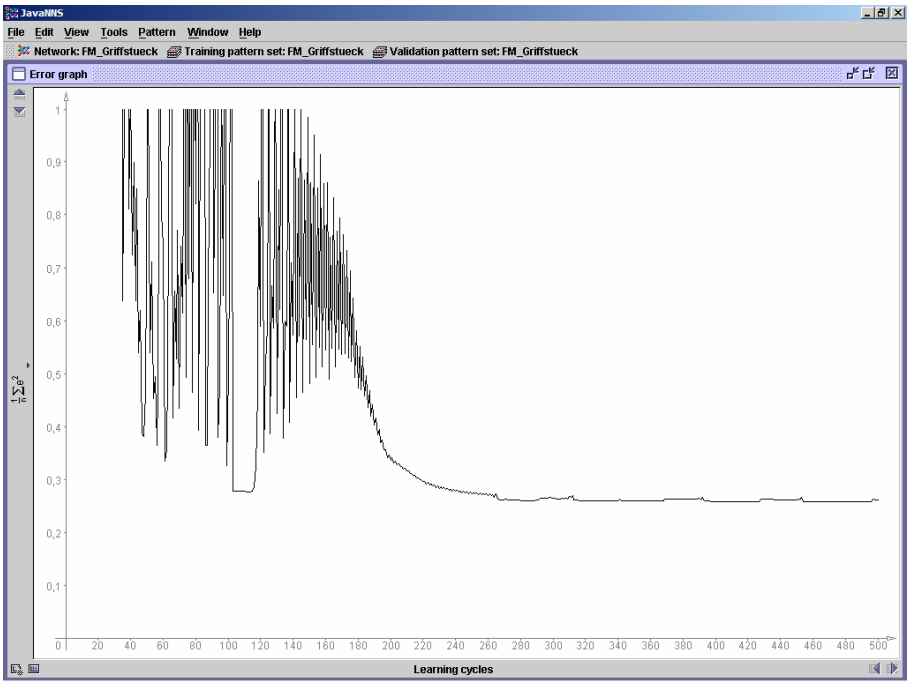

Figure 7. Failure trend for backpropagation learning with three hidden neurons

The extreme reduction of hidden neurons processed here was only possible because of the relatively low quantity of training data sets. Although in real product application this will rarely occur, the proceeding was appropriate to explain the general process and the possibilities of optimization.

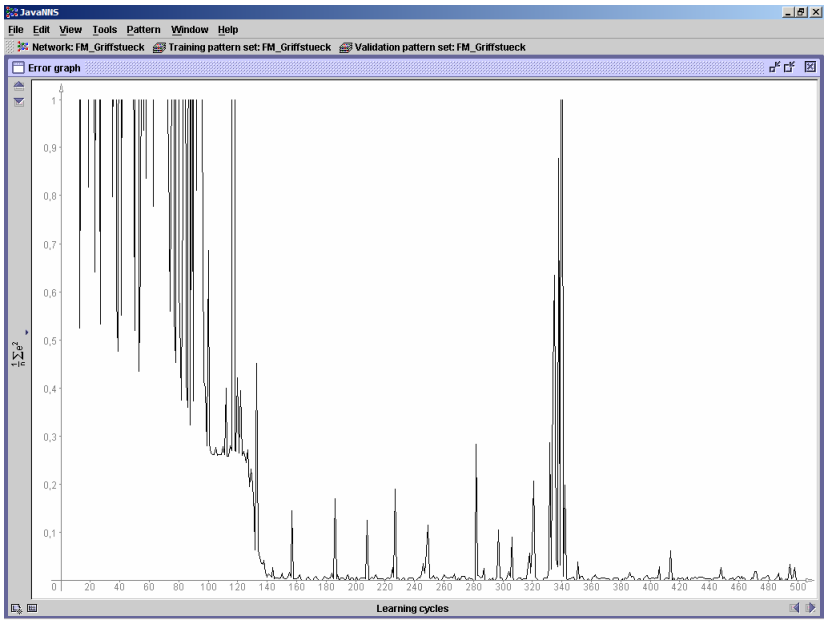

Figure 8. Failure trend for Quickprop learning with two hidden neurons

Three further modules are required beside of the network implementation itself, if the shown network approach is planned to be used for practical product specification: input module, output module, and regulator. By applying the input module, the user can insert input data to the network. Such a module can e.g. be realized by use of computer supported user forms or data base connections, depending on the specific use case.

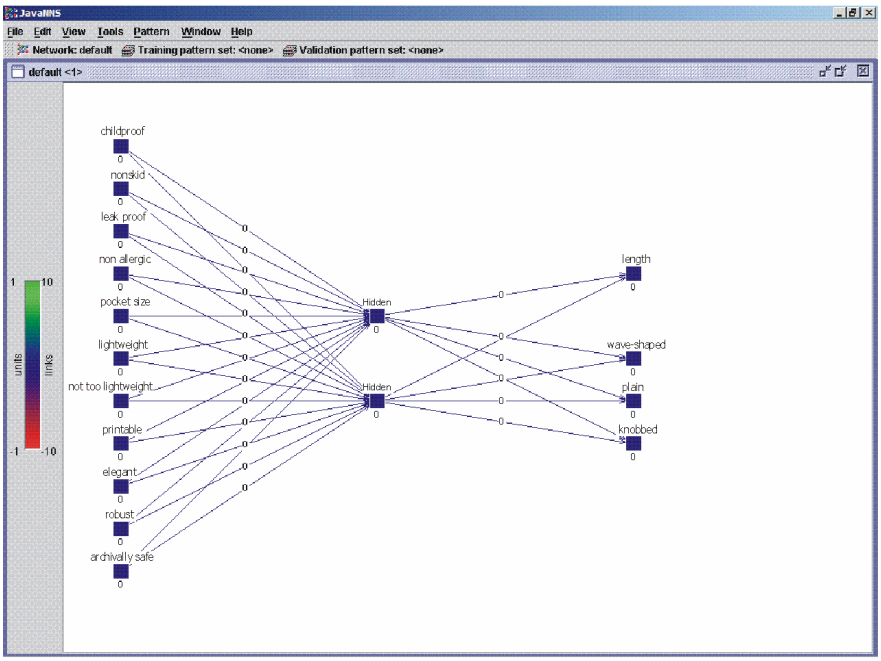

Figure 9. Finally obtained neural network structure

The output module mediates the results to the user, e.g. as a short summary of the resulting product specification (“transparent plastic ball pen with fixed ink store and plain grip”) or in form of a bill of materials with detailed parts, their dimensions and materials. If powerful system integration can be realized, the automated generation of a CAD model is possible. The regulator is required for the last two examples of output representation. This module contains information about the product spectrum parts and their properties as well as the rules for

determining these properties by the available network output. The required rules are derived from the process of output definition.

4 Transfer to general product specification

The presented approach shows the specification of customer induced individualized products by extraction from comprehensive product spectra. Procedures are kept on an abstract level to permit easy transfer to further problem areas. Real world problems usually possess a distinctly higher degree of freedom than the shown example of ball pen specification. Thus, it is important to attach importance to the correct execution and the use of available tools to manage complexity. The core of the approach is the creation of problem descriptions, as all further steps of the neural network design are based on this. The explained steps for determining input and output data as well as the dependencies between different output data represent a useful guideline. Product data is often available in digital form, which should be consequently used. Furthermore, the obtained information is to be documented, e.g. in computer based matrices or graphs. Especially when addressing implicitly available product data, the acquisition in interdisciplinary teams should be preferred.

The efficiency of the created neural network largely depends on the quality of collected training data. In real world problems, a simple collection often is not possible, as it has been done for the ball pen example. Customers often are not able to carry out the product specifications by themselves, especially when referring to complex products. For innovative products a further difficulty is the fact that customers are even unable to define their requirements. In this case, indirect strategies and methods (e.g. prognosis) are required as well as intensive support from marketing and distribution experts. For design and implementation of neural networks, a simple guideline is presented as well. When considering complex problems the possibility to subdivide neural networks in order to facilitate their design and computing is a useful method to keep structures manageable. Due to large differences in possible problem situations, a universal optimization approach cannot be indicated. For practical optimization, it is promising to adapt the neurons in the hidden layer depending on the resulting failure trends and to comparatively adopt available basic learning procedures.

5 Conclusions

Neural networks provide a powerful possibility to describe and adapt product models without requiring detailed analytical knowledge about them. Therefore, these networks are well suited for customizing products with respect to individual customer requirements. We present a procedure for creating a problem specific neural network, which is modularized in five main steps. The proceeding offers new possibilities for individual product adaptation, which cannot be provided by conventional methods focusing on product selection. Due to a generalized description, the approach is easy to apply to different products. Further research will focus on adaptation processes, which are initialized by conclusions drawn from neural network responses.

6 Acknowledgments

We would like to thank the DFG (Deutsche Forschungsgemeinschaft) for funding this project as part of the SFB 582, as well as our partners from the industry for good collaboration and extensive support.

References

[1] Pulm, U., Maurer, M., Lindemann, U.: Methods, Tools, and Processes for the Design and Development of Individualised Products, Proceedings of the ICED 03, Stockholm, 2003, pp 477-484.

[2] Ericsson A.;Erixon G.: Controlling Design Variants – Modular Product Platforms. ASME Press: New York 1999.

[3] Patterson, D.: Artificial Neural Networks - Theory and Applications., Singapore: Prentice Hall 1996.

[4] Rojas, R.: Neural Networks - A Systematic Introduction. Berlin: Springer 2002.

[5] Hubka, V.; Eder, W.: Theory of Technical Systems. Berlin: Springer 1988.

[6] Pahl, G.; Beitz, W.: Engineering Design. A Systematic Approach. London: Springer 1996.

[7] Kusiak, A.: Computational Intelligence in Design and Manufacturing. New York: John Wiley & Sons 2000.

[8] Zell, A.; Mamier, G.; Vogt, M.; et al. SNNS – User Manual, Version 4.2 URL: http://www-ra.informatik.uni-tuebingen.de/SNNS/, last access 2005-03-08.

[9] Fischer, I.; Hennecke, F.; Bannes, C.; Zell, A. JavaNNS – User Manual, Version 1.1 URL: http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/, last access 2005-03-08.

[10] Raudys, S.: Statistical and Neural Classifiers - An Integrated Approach to Design. London: Springer 2001.

Maik Maurer
Institute of Product Development
Technische Universitaet Muenchen
Boltzmannstr. 15
D-85748 Garching
Germany
Tel: +49 (0)89 289 15155
Fax: +49 (0)89 289 15144
E-mail: maurer@pe.mw.tum.de