A HYBRID KNOWLEDGE PROCESSING ARCHITECTURE

U. Rückert

Technical University of Hamburg-Harburg, Germany

Abstract

An example of a hybrid system architecture integrating connectionist models and symbolic knowledge processing is introduced. In particular the inclusion of non-symbolic knowledge sources (data from physical processes, measurements, sensors etc.), transformation of learned knowledge into a symbolic form (rule extraction) and associative storage as well as retrieval of information will be discussed by means of application examples. Finally, parallel hardware support for the proposed hybrid system architecture will be presented.

INTRODUCTION

In recent years there has been an increasing interest in alternative approaches to information processing systems in those domains where at present humans outperform any currently available high performance computers. Examples may be seen in areas like auditory perception, vision, or sensory-motor control. In principle these approaches can be classified as *symbolic* or *sub-symbolic* data processing systems and the combination of both paradigms (Palm et. al. (1)).

The usefulness of a symbolic knowledge processing and representation on which traditional artificial intelligence (AI) relies has been shown in areas like diagnosis, construction, and planning. An important property of knowledge stored in symbolic form is that it can be interpreted and communicated by experts or computers. The limits of such an approach, however, turn out to be quite evident when sensor data or measurement data are handled. Inconsistent data can force symbolic systems into an undefined state. A further problem in expert system design is the acquisition of knowledge. It is almost impossible for an expert to describe his domain specific knowledge entirely in form of rules or other knowledge representation schemes. Hence, automatic knowledge acquisition (machine learning) methods are of great interest.

Known for their learning capabilities are artificial neural networks (ANNs). They are successful in technical applications dealing with *sub-symbolic raw data*, in particular, if the data are noisy or inconsistent. The computing power of biological neural networks stems to a large extend from a highly parallel, fine-grained and distributed processing and storage of information as well as from the capability of learning. Such subsymbolic-level processing seems to be appropriate for perception tasks and perhaps even for tasks that call for combined perception and cognition. For example, ANNs are able to learn structures of an input set without using a priori information. Unfortunately, they cannot easily explain their behaviour because a distributed representation of the knowledge is used. Typically, they only can tell about their knowledge by showing responses to given inputs. In other words, they have no reasoning component. Despite this disadvantage, neural information processing is expected to have a wide applicability in areas that require a high degree of flexibility and the ability to operate in uncertain environments where information usually is partial, fuzzy, or even contradictory.

Both approaches of modelling brain like information processing capabilities are complementary in the sense that traditional AI is a top-down approach starting from high-level cognitive brain functions whereas ANNs are a bottom-up approach on a biophysical basis of neurons and synapses (1). It is a matter of fact that the symbolic as well as the subsymbolic aspects of information processing are essential to systems dealing with real-world tasks. When we try to recognize a certain pattern, it helps if we know what we are looking for (Pao (2)). The combination of both paradigms allows the merging of learning algorithms offered by ANNs and the representation of qualitative transparent rules in inference systems. Hence, linking symbolic and subsymbolic information processing is certainly a challenging research task motivating a national research project sponsored by the german government. The project, called *Knowledge Processing in Neural Architecture*, started at the beginning of 1991 and will be finished at the end of 1994. The partners of the project are the University of Ulm, Dept. of Neural Information Processing, headed by Prof. Palm, the University of Dortmund, Dept. of Electrical Engineering (Prof. Goser), the University of Marburg, Dept. of Artificial Intelligence (Prof. Ultsch) and the Technical University of Hamburg-Harburg, Dept. of Technical Electronics (Prof. Rückert). Scientific objectives and first results of our project will be presented in the following.

'Intelligent Systems Engineering', 5–9 September 1994, Conference Publication No. 395, © IEE, 1994

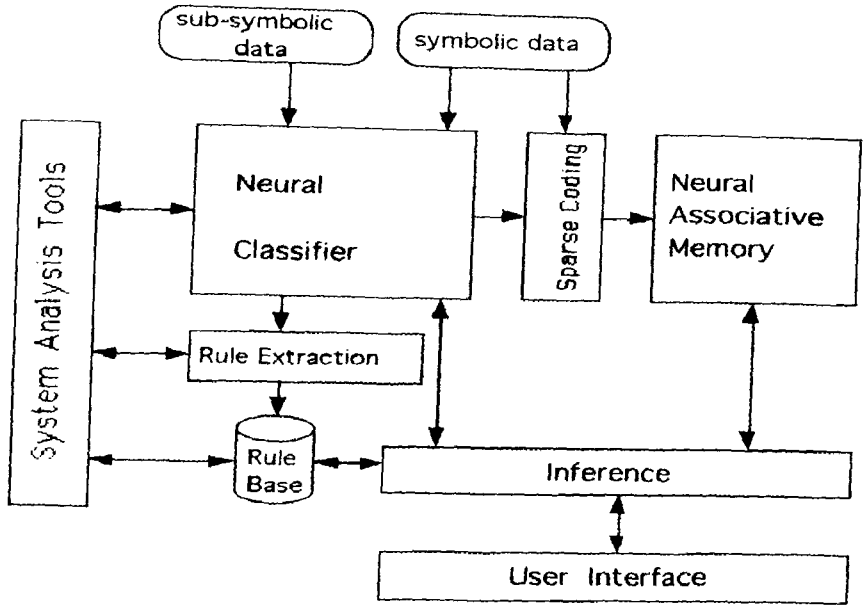

Figure 1: Architecture of the hybrid knowledge processing system

ARCHITECTURE OF THE HYBRID SYSTEM

An overview of the hybrid knowledge processing system developed in the project is given in Fig. 1. To be able to meet the demands of many real-world problems the system can deal with symbolic as well as sub-symbolic input data, possibly from different sources. At first, the input data has to be analysed in order to find a convenient and valid organization of the data, based on the inherent structure and relationships among the patterns. This process may be seen as dividing the pattern space into a minimal number of regions over which this pattern space is relatively uniform on the basis of perceived similarities and assigning them to different classes. Especially for large high-dimensional data spaces this processing calls for efficient and robust data analysis techniques. Some methods are known from multivariate statistics (e.g. cluster analysis) and pattern recognition (Bezdek and Pal (3)). Furthermore, ANNs have been proposed by several authors for this task as well (e.g. (2) and (3)), since ANNs are claimed to have advantages over conventional methods in handling noisy and incomplete data. In addition, by utilizing the parallelism inherent in ANNs an efficient implementation for real-time applications is feasible.

Within our hybrid system data analysis is done by the module called *Neural Classifier* (Fig. 1). At present, we are mainly engaged in *selforganizing feature maps* (SOFMs) as proposed by Kohonen (4) and variants of them (e.g. (5), (6)). A SOFM uses an unsupervised learning algorithm to adapt itself suitably to the structure of a given (high dimensional) data space. The algorithm can be thought of as a mapping from R^n to a flattened two-dimensional surface (layer of neurons) such that interesting topological relations and the point density of the vectors are preserved (4). For the interpretation of the learning result of SOFMs an automatic method for visualization of clusters is required. Such a method was developed by Ultsch and Siemon (7), the so called *unified distance matrix* or short *U-matrix method*. The U-matrix displays its elements as hight over a grid that corresponds to the lattice of the SOFM. This display has valleys where the weight vectors in the map are similar and hills or walls where the weight vectors in the map have large differences. Consequently, groups of neurons on the SOFM representing data sets that have something in common are separated by walls. A wall represents large weight vector to weight vector distances of adjacent neurons and indicates the dissimilarity of the vectors in the data space. For a more detailed discussion of this method see (7).

By using SOFMs and the U-matrix method structure in data sets can be detected and visualized automatically. The next step is the automatic interpretation of the results which is the task of the *rule extraction* module (Fig. 1). This module aims at automatic discovery of the properties of each cluster detected by the neural classifier and their reformulation into a symbolic form. Therefore, the structure learned by the ANN has to be examined and subsequently transformed into (e.g. PROLOG) rules. These rules can then be inspected by a human expert and added to a rule base.

Two different methods have been implemented within our project so far. The first algorithm called *sig** developed by the group of Ultsch (7) is able to generate PROLOG rules whereas the second algorithm by Surmann *et al* (8) generates fuzzy rules. Both approaches are based on the clustering result of a SOFM and the resulting U-matrix. By inspecting the individual components of the neuron weight vectors the

implemented methods try to discover algorithmically the properties of the clusters. For example, by taking a look at the weight vector components of the trained SOFM it is possible to formulate simple fuzzy rules:

If comp0 is high and comp1 is low and comp2 is low then vector belongs to cluster2
If comp0 is high and comp1 is high and comp2 is medium then vector belongs to cluster4
....

Obviously, the implementation of the above mentioned intuitive operation *by looking at the component cards* is not straightforward. The proposed algorithms are now tested and evaluated with real-life application data in more than three dimensions. An example is given later on (Applications).

In summary, one important path in our hybrid knowledge processing system is from sub-symbolic and symbolic data sources via the neural classifier and the rule extraction module to the rule base of the inference mechanism (Fig. 1). By using ANNs this approach enhances the reasoning capability of classical expert systems with the ability of generalization and the handling of incomplete data. At present, the inference module consists of a commercially available PROLOG interpreter for symbolic proofs and a simple fuzzy controller. Since there is still a lack of understanding in all these steps of this path, analysis tools for each of the subcomponents are an important part of the system (*system analysis tools*, Fig. 1). For example, to explore the properties of a SOFM we already proposed to interprete the map by means of the U-matrix. Alternative graphical representations are *spanning trees, vector maps*, and *component cards* (4). Examples for analytic evaluation parameters are homogeneity or heterogeneity of the learned clusters or the topographic product (9). The main motivation for using SOFMs for data analysis is the possibility to extract automatically the learned knowledge which is not the case for many other ANN models. The SOFM have similarities to fuzzy and adaptive k-means clustering (3), combined with a multidimensional scaling algorithm to get a low dimensional representation of the cluster centres. But for SOFMs we don't need to fix the number k of cluster centres. A disadvantage of the SOFMs is the lack of a permanent learning capability. So far, the training process of the SOFM is a batch process which is not appropriate for adaptive systems. Therefore, several variants of the original SOFM algorithm are under development at the moment (see e.g. (5), (6)).

The other important path of our system goes from the Neural Classifier through the *Neural Associative Memory (NAM)* to the inference modul. NAMs for which theoretical work on their performance is already elaborated are used for two types of tasks: fault tolerant pattern mapping and pattern completion (e.g. (4) and Palm (10)). ANNs are well suited for the implementation of associative memories, at least because the processing elements (artificial neurons) in an ANN operate in a highly parallel way and thus a considerable gain in speed is to be expected (Rückert (11)). Another interesting feature of NAMs is that information is stored in a distributed way over many processing units and not anywhere in particular. The distributed representation of information is a major characteristic of NAMs and appears to be particular appropriate for massively parallel systems. Obviously, this functionality makes them attractive for the use in our hybrid knowledge processing system. Especially, in regard to noisy or incomplete inputs of subsymbolic and symbolic data sources, because the access to stored data is based on similarity.

The NAM path is mainly used for fast and fault-tolerant access to stored factual knowledge (facts) supplied by an expert or extracted from the input data (Neural Classifier). An incoming pattern (question, stimuli) would be classified by the Neural Classifier and passed to the NAM which in turn associates the corresponding output pattern (answer, reaction). If the associated output is questionable, the inference module is evoked to resolve the conflict. Another possibility is that the inference module controls the interaction between the Neural Classifier, the NAM and itself. A query from the user via the user interface (Fig. 1) is processed by the inference module which apply partially instantiated facts to the NAM when necessary, and the NAM returns the best match from stored facts. The inference module also has access to the Neural Classifier; either to obtain detailed information on the input data, or to provide additional input for the classification of those data. This can be important if there already exists knowledge about structures in the input data, which might be used during the classification process (1).

One important condition for an efficient use of NAMs is a *sparse coding* of input/output patterns. In most ANN models for associative memory the problem of coding of the input and output patterns is not explicitly discussed. Usually, randomly generated patterns are assumed with a certain probability p for a component to be active ('1') and a corresponding probability 1-p for a component to be inactive ('0'). Most of the models only consider p=1/2. But it turns out, that for sparsely coded patterns (small p) the storage efficiency of NAMs (number of bits per synapse that can be stored) and the number of patterns that can be stored with low error probability is much larger (10). The problem of sparse coding of I/O patterns is one of the basic problems that has to be considered for each prospective application of a NAM. The investigation of various data analysis techniques (see above) helps in designing sparse codes. For example, a cluster analysis groups similar objects into (usually disjoint) subsets called *cluster*. The membership of a pattern to a cluster may be interpreted as a special property of this pattern which can be used to transform the pattern into a sparsely coded feature vector. In addition to sparseness, such a code may have

the further important advantage of being similarity preserving (Palm (12)).

The research group of Palm (Ulm) is actually working on a theory and on formal design strategies for sparse, similarity preserving codes. The research group of Rückert (Hamburg-Harburg) explores the use of trained SOFMs for the generation of such codes. Even if the problem of sparse coding has not been solved completely at the moment, such a code is biologically motivated (12) and has a great deal of potential in associative storage and retrieval of information.

PROTOTYPE IMPLEMENTATION

Apart from the theoretical investigations the implementation of the hybride system (Fig. 1) was another important topic of our research project. A realization of a prototype system has a couple of interesting consequences. For example, interfaces between the different components have to be developed in order to get a working prototype. Theoretical results concerning the application of different ANNs for certain subtasks within such a system could be verified directly. Last but not least, test applications of the prototype system will give an idea of the computational requirements of the system which may motivate parallel hardware implementation of neural subcomponents.

The most flexible approach is the software implementation on a workstation. Each component of the hybrid knowledge processing system has been individually implemented and tested. In addition to the methods discussed above, alternative methods have been implemented for each module. For example, for the Neural Classifier different variants of SOFMs and an adaptive k-means clustering algorithm have been implemented. Alternatives for the NAM are b-trees and hashing as well as variants of NAMs. Together, all modules build up a working and flexible software prototype of the desired hybrid system.

Simulation of ANNs on conventional (serial) hardware is rather slow, especially for large net sizes. Therefore, parallel hardware support is greatly appreciated. Within our project we developed a distributed and hybrid parallel hardware support based on the VME-Bus. The system can be connected to a SUN workstation or a PC and integrates commercially available processor cards as well as special-purpose add-on-boards for fast simulation of ANNs. For NAMs with binary weights and input/output patterns a special purpose SIMD architecture called PAN IV (Parallel Associative Network) was built (Palm and Palm (13)). The first prototype system consists of 144 special purpose ICs (digital ASICs) and 144 MByte memory. For Neural Classifiers and variants of the binary NAM an add-on-board on the basis of the INTEL i860 microprocessor was developed. Both subsystems were designed for the VME-Bus.

On our hardware platform the concept of *virtual networks* builds the fundamental base for multiuser/multitasking operation controlled by the operating system PANOS (Parallel Associative Network Operating System, (13)). Several *virtual networks* can be defined and placed arbitrarily into the physical network provided by the hardware. Input/Output control is done by the workstation.

In addition to these parallel hardware platforms task-dedicated VLSI architectures for NAMs and SOFMs have been implemented as well. At the University of Dortmund and Technical University of Hamburg-Harburg(Goser/Rückert) several prototype chips for NAMs based on analog, digital and digital/analog circuit techniques have been successfully realized and tested already (Goser *et al* (14), Rückert (11)). A digital VLSI-chip was implemented for SOFMs (Rüping *et al* (15)). Based on these VLSI architectures for NAMs and SOFMs which are optimized in respect to speed, size and testability a small size and high performance microsystem implementation of our hybrid knowledge processing system is feasible.

APPLICATIONS

The four research groups cooperating in the WINA project are investigating different applications of the hybrid knowledge processing system. The work on these applications is done in close cooperation with industrial partners and other faculties at the Universities of Dortmund, Hamburg-Harburg, Marburg, and Ulm. The Palm group (Ulm) works on speech recognition (spoken words or syllables, subsymbolic data) and on written text (symbolic data). By using the PAN IV hardware they develop a new associative information-retrieval system which is able to deal with complex subsymbolic and symbolic data structures. The Goser (Dortmund) group applies the hybrid knowledge processing system to process monitoring and quality assurance of their own chip fabrication facilities.

The Ultsch group (Marburg) is engaged with medical (blood) and environmental (local water quality) data for diagnostic purposes. For example, they tested the path from sub-symbolic inputs via a neural classifier to the rule-base with a data set of 242 patients with 11 clinical test values each (7). This data set was used for diagnosis of iron defficiency. The rules generated by the sig* algorithm (rule extraction module) show a high degree of coincidence with expert's diagnosis rules, and exhibit knowledge not prior known to them while making sense to the expert (16). Deviding the data set equally into a training set and a test set the system produced in 119 out of 121 cases (98%) the right diagnostic (7). Hence, the hybrid system implements

the automatic acqusition of knowledge out of a set of unlabeled examples very well. In addition, this group applies the hybrid knowledge processing system to quality assurance (mechanical engineering) and process control.

The Rückert group (Hamburg-Harburg) implemented a document retrieval system based on the hybrid knowledge processing architecture. The system was tested with artificial test databases of up to 300.000 keywords and references. This application makes use of the path from symbolic inputs (typed keywords) via the Neural Classifier (SOFM), the Neural Associative Memory, and the inference module to the user interface. A SOFM (70x70 neurons) was trained with a set of about 8000 different trigrams which have been extracted from all used keywords. Trigrams are sequences of three consequtive letters. For example, the keyword "neural" includes the following trigrams, whereby the "#" character marks the beginning and end of the keyword: neural = (#ne, neu, eur, ura, ral, al#).

Given a keyword as input, the corresponding trigrams are extracted and passed sequentially to the SOFM. The best matching neuron for each trigram of the input keyword is marked as a point on the SOFM. Connecting these points with their successors by a straight line implies a characteristic trajectory for this keyword on the two dimensional map as show in Fig. 2. The result of this keyword preprocessing is a sparsely coded binary vector with 4900 components in which only h components are set to '1'. The number h equals the number of trigrams or letters in the input keyword. The activated components ('1') correspond to the position of the best matching neurons in the SOFM. Obviously, the use of a SOFM for this kind of sparse coding based on trigrams is somewhat artificial. The proposed coding could be done directly with the help of a simple coding algorithm (Heimann (17)) resulting in a slightly better performance. But it shows that the application of SOFMs is not restricted to subsymbolic data and it points out the universality of our hybrid system architecture.

The sparsely coded binary representation of each keyword is stored together with a sparsely coded address to an external list of associated references. In other words the NAM implements a so called *inverted list* (17). The address vector has 4000 components with k=4 activated components ('1'). This format in combination with a simple decoding algorithm can be easily transformed into a decimal notion of the address (17). Because the output of a NAM may contain more than the required k activated components (10), the decoding is done with the help of the inference module containing knowledge about the decoding algorithm and rules for resolving conflicts in the output vector (additional '1's).
At first glance, this architecture seems to be too complicated for such a simple task of mapping keywords to addresses of reference lists. Especially,

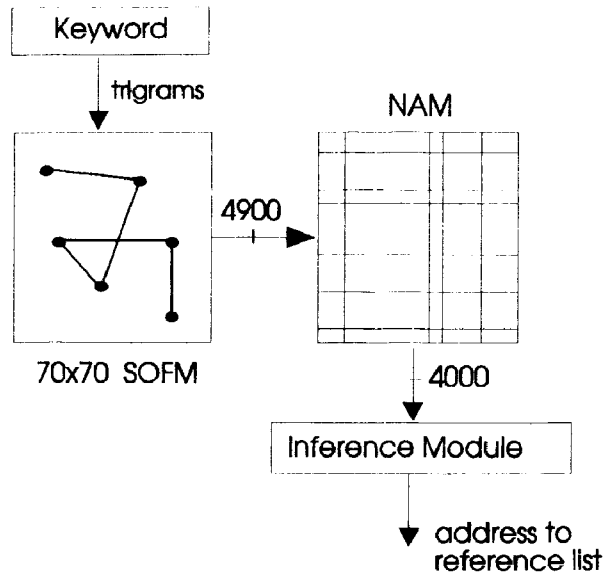

Figure 2: The hybrid system architecture for document retrieval

because from computer science very efficient algorithms are known like hashing and b-trees (Aoe (18)). But a comparison shows that the performance of these different methods is similar. In Table 1 the result of such a comparison for a data set of about 175,000 pattern pairs (keyword, address) is summarized. All methods have been implemented on the same workstation (Sparc 2) under the same software environment. As can be seen from Tab. 1, the presented hybrid system requires the smallest amount of memory compared to hashing and b-trees. This is suprising, in particular because the applied sparse input/output coding is not optimal in respect to the storage efficiency of the NAM (17). On the other hand the access time which is the time for calculating the associated address to the reference list (in decimal notion) is longer. This is not suprising, because hashing and b-trees are methods optimized for sequential computing machines, whereas ANNs are inherently parallel algorithms. Consequently, on the basis of adequate parallel hardware for the ANN modules (see Prototype Implementation) the access time of the hybrid approach will be speeded up considerably. Last but not least, the hybrid system shows to a certain degree fault-tolerance, which means that missing characters, additional characters, or switched adjacent characters in the keyword will not seriously affect the performance. Unfortunately, we are not able to quantify this fault-tolerance of our system at the moment. We are still working on this topic.

Table 1 Comparison of different key search strategies [18]

method	memory usage [MB]	access time [ms]	parallelism	fault-tolerance
b-tree	5.2	0.1	-	-
hashing	4.0	0.3	-	-
hybrid system	2.0	1.2	++	+

CONCLUSION

The presented national research project *Knowledge Processing in Neural Architecture* aims at combining neural information processing and methods known from Artificial Intelligence. ANNs may bridge the gap between the 'subsymbolic' raw data and symbolic knowledge processing. The hybrid knowledge processing system proposed in this paper is one of the possible ways to combine the advantages of the symbolic and subsymbolic paradigms. The presented system architecture is modular and transparent, so that the extension with additional modules or the exchange of single modules can be done easily. Furthermore, the software implementation of ANN modules can be speeded up by dedicated hardware implementations. Our first results show that the combination of both approaches is not only feasible but also useful, even though our approach was not driven by a special application.

ACKNOWLEDGEMENT

This work has been supported by the german ministry of research and technology BMFT, contract number 01-IN 103 B/O.

REFERENCES

1. Palm, G., Goser, K., Rückert, U. and Ultsch, A., 1992, Proc. of the Int. Workshop on VLSI for Neural Netzworks and Artificial Intelligence, Oxford, UK, J3

2. Pao, Y.H., 1989, "Adaptive Pattern Recognition and Neural Networks", Addison Wesley, Reading Massachusetts

3. Bezdek, J. C., and Pal, S.K., 1992,"Fuzzy Models For Pattern Recognition", IEEE Press, New York

4. Kohonen, T., 1984, "Self-organization and associative memory", Springer, Heidelberg

5. Fritzke, B., 1991, Proc. of IJCNN-91, 531-536

6. Martinetz, T., and Schulten, K., 1994, Neural Networks, 7, 507-522

7. Ultsch, A., 1992, Proc. of the ECAI 92. 10th European Conference on Artificial Intelligence, 208-210

8. Surmann, H., Möller, B, and Goser, K., 1992, Proc. of the 5th Int. Conf. on Neural Networks & their Applications, NEURO NIMES 92, 187-194

9. Bauer, H.-K., and Pawelzik, K., K., 1992, IEEE Trans. on NN, 3 (4), 570-579

10. Palm, G., 1980, Biol. Cybern., 36, 19-31

11. Rückert, U., 1991, Proc. of the 24. HICSS, Hawaii, USA, 212-218

12. Palm, G., 1991, Concepts in Neuroscience, 2 (1), 97-128

13. Palm, G., and Palm, M., 1991, Proc. of the 2nd Int. Conf. Microelectronics for Neural Networks, Kyrill&Method Verlag, München , Germany, 411-416.

14. Goser, K., Hilleringmann, U., Rückert, U., and Schumacher, K., 1989, IEEE Micro, 9, 28-44

15. Rüping, S., Rückert, U., and Goser, K., 1993, Proc. IWANN, Sitges, Spain, 488-493

16. Ultsch, A., 1991, Research Report No. 396, University of Dortmund, Dept. of Computer Science (in german)

17. Heimann, D., 1994, PhD thesis, Technical University of Hamburg-Harburg, Dept. of Technical Electronics (in german)

18. Aoe, J. K, 1991, "Computer Algorithms: Key Search Strategies", IEEE Computer Society Press.