Integration of Neural Networks with Knowledge-Based Systems

Alfred Ultsch, Dieter Korus
Department of Mathematics/Informatics; University of Marburg
Hans-Meerwein-Straße/Lahnberge; D-35032 Marburg; F. R. Germany
email: ultsch or korus@mathematik.uni-marburg.de
http://www.uni-marburg.de/~wina/

ABSTRACT

Existing prejudices of some Artificial Intelligence researchers against neural networks are hard to break. One of their most important arguments is that neural networks are not able to explain their decisions. Further they claim that neural networks are not able so solve the variable binding problem for unification. We show in this paper that neural networks and knowledge-based systems must not be competitive, but are capable to complete each other. The disadvantages of the one paradigm are the advantages of the other and vice versa. We show several ways to integrate both paradigms in the areas of explorative data analysis, knowledge acquisition, introspection, and unification. Our approach to such hybrid systems has been proven in real world applications.

1. Introduction

The successful application of knowledge-based systems in different areas as diagnosis, construction and planning shows the usefulness of a symbolic knowledge representation. However, this representation implies problems in processing data from natural processes. Normally such data are results of measurements and have therefore no straightforward kind of symbolic representation [1]. Knowledge-based systems often fall short in handling inconsistent and noisy data. It is also difficult to formalize knowledge in such domains where ‘a priori’ rules are unknown. Often the performance in ‘learning from examples’ and ‘dealing with untypical situations’ (graceful degradation) is insufficient. The rules used by conventional expert systems are said to be able to represent complex concepts only approximately [4]. In such complex systems inconsistent and context-dependent rules (cases) may result in unacceptable errors. In addition, it is almost impossible for experts to describe their knowledge, which they acquired from many examples by experience, entirely in symbolic form [6].

State-of-the-art knowledge-based system technology is based on symbolic processing. Acknowledged shortcomings of current computational techniques is their brittleness, often arising from the inability of first order logic to capture adequately the dynamics of a changing and incompletely known environment. An important property of knowledge stored in symbolic form is that it can be interpreted and communicated to experts. The limits of such an approach, however, become quite evident when sensor data or measurement data, for example from physical processes, are handled. Inconsistent data frequently force symbolic systems into an undefined state.

Another heavy problem in knowledge-based system design is the acquisition of knowledge. It is well known that it is almost impossible for an expert to describe his domain specific knowledge entirely in form of rules or other knowledge representation schemes. In addition, it is very difficult or even impossible to describe expertise acquired by experience.

Neural networks claim to avoid most of the disadvantages of knowledge-based systems described above. These systems which rely on a distributed knowledge representation are able to develop a concise representation of complex concepts. It is possible to learn knowledge from experience directly [4]. Characteristic attributes of connectionist systems are the ability of generalization and graceful degradation. E.g. they are able to process inconsistent and noisy data. In addition, neural networks compute the most plausible output to each input. Neural networks, however, also have their disadvantages. It is difficult to provide an explanation of the behaviour of the neural network because of the distributed knowledge representation. Therefore expertise learned by neural networks is not available in a form that is intelligible for human beings as well as for knowledge-based
systems. It seems to be difficult to describe or to interpret this kind of information. In knowledge-based systems on the other hand it is easy to describe and to verify the underlying concepts.

2. Integration of Neural Networks with Knowledge-Based Systems

Indications are that neural networks provide fault-tolerance and noise resistance. They adapt to unstable and largely unknown environments as well. Their weakness lies in a reliance on data-intensive training algorithms, with little opportunity to integrate available, discrete knowledge. At present, neural networks are relatively successful in applications dealing with subsymbolic raw data; in particular, if the data is noisy or inconsistent. Such subsymbolic level processing seems to be appropriate for dealing with perceptions tasks and perhaps even with tasks that call for combined perception and cognition. Neural networks are able to learn structures of an input set without using a priori information. Unfortunately they cannot explain their behavior because a distributed representation of the knowledge is used. They only can tell about the knowledge by showing responses to a given input.

Both approaches, knowledge-based systems and neural networks, of modelling brain-like information processing are complementary in the sense that traditional knowledge-based systems are a top-down approach starting from high-level cognitive functions whereas neural networks are a bottom-up approach on a biophysical basis of neurons and synapses. 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. Integrating neural networks and knowledge-based systems is certainly a challenging task [10]. Beside these general considerations several specific tasks have to be solved. The most important are - without claiming on completeness:

Structure Detection by Collective Behavior: In real world people have continuously to do with raw and subsymbolic data which is characterized by the property that one single element does not have a meaning (interpretation) of itself alone. The question is, how to transform the subsymbolic data into a symbolic form. Unsupervised learning neural networks can adapt to structures inherent in the data. They exhibit the property to produce their structure during learning by the integration (overlay) of many case data. But they have the disadvantage that they cannot be interpreted by looking at the activity or weights of single neurons. Because of this we need tools to detect the structure in large neural networks.

Integrated Knowledge Acquisition: Knowledge acquisition is one of the biggest problems in artificial intelligence. A knowledge-based system may therefore not be able to diagnose a case which an expert is able to. The question is, how to extract experience from a set of examples for the use of knowledge-based systems. Under Integrated Knowledge Acquisition we understand subsymbolic approaches, i.e. the usage of neural networks, to gain symbolic knowledge. Neural networks can easily process subsymbolic raw data by handling noisy and inconsistent data. An intrinsic property of neural networks is, however, that no high level knowledge can be identified in the trained neural network. The central problem for Integrated Knowledge Acquisition is therefore how to transform whatever a neural network has learned into a symbolic form.

Introspection: Under introspection we understand methods and techniques whereby a knowledge-based system observes its own behaviour and improves its performance. This approach can be realized using neural networks that observe the sequence of steps an expert system takes in the derivation of a conclusion. This is often called control knowledge. When the observed behaviour of the expert system is appropriately encoded, a neural network can learn how to avoid misleading paths and how to arrive faster at its conclusions.

Unification: One type of integrated reasoning is the realization of an important part of the reasoning process, the unification, using neural networks. Unification pays a central role in logic programming (e.g. in the language Prolog) and is also a central feature for the implementation of many knowledge-based systems. The idea of this approach is to realize the matching and unification part of the reasoning process in a suitable neural network.

3. Structure Detection by Collective Behavior

One of the neural network types we use for representing subsymbolic raw data in large distributed neural networks are the Self-Organizing Feature Maps (SOFM) by Kohonen [5]. It has the ability to map a high-dimensional feature space onto a usually two-dimensional grid of neurons. The important feature of this mapping is that adjacent points in the data space are mapped onto adjacent neurons in the grid by conserving the distribution
of the input data. In normal applications we use 64 by 64, 128 by 128 or 256 by 256 neurons. Due to the representation of the input data in learning phase the SOFM adapts to the structure inherent in the data. On the map neighbouring neurons form regions, which correspond to similar input vectors. These neighbourhoods form disjoint regions, thus classifying the input vectors.

But looking at the learned SOFM as it is one is not able to see much structure in the neural network, especially when processing a large amount of data with high dimensionality. In addition, automatic detection of the classification is difficult because the SOFM converges to an equal distribution of the neurons on the map. So a special visualization tool, the so called „unified distance matrix methods“, short U-matrix methods, were developed [19] to graphically visualize the structure of the SOFM in a three-dimensional landscape (fig. 1). The simplest U-matrix method is to calculate for each neuron the mean of the distances to its (at most) 8 neighbours and add this value as the height of each neuron in a third dimension. Other methods e.g. consider also the position of the reference vectors on the map. Using an U-matrix method we get, with the help of interpolation and other visualisation technics, a three-dimensional landscape with walls and valleys. Neurons which belong to the same valley are quite similar and may belong to the same class; walls separate different classes (fig. 1). Unlike in other classification algorithms the number of expected classes must not be known a priori. Also, subclasses of larger classes can be detected. Single neurons in deep valleys indicate possible outliers. These visualizations are implemented together with a complete toolbox to show the interpolated U-matrices in three dimensions, with different interpolation methods, in different colour tables, different perspectives, with clipping, tiled or single top view, the position of the reference vectors, the identification of single reference vectors to identify possible outliers or special cases, the drawing of class borders, labeling of clusters, and in addition, single component maps, which show the distribution of a single feature on the SOFM. For example, using a data set containing blood analysis values from 20 patients (20 vectors with 11 real-valued components) selected from a set of 1500 patients [3], it turned out that the clustering corresponds nicely with the different patient's diagnoses.

4. Integrated Knowledge Acquisition

In the previous Section we presented the combination of the Self-organization Feature Map (SOFM) by Kohonen [5] and the U-matrix methods [19] to detect structure in large neural networks with collective behaviour representing the structure of the input data. As a result we are able to classify the input data. To acquire knowledge out of this neuronal classification, we developed an inductive machine learning algorithm, called sig*.
Fuzzy logic, based on fuzzy set theory [20], opens the possibility to model and process vague knowledge in knowledge-based systems. This offers for example the chance to explain the decision making process of human experts derived from vague or uncertain information. In another way some problems of traditional knowledge-based systems like dealing with exceptions in rule based systems can solved by fuzzy logic. Further, because of the generalisation ability of neural networks fuzzy theory is well suited to express the vague knowledge of learned neural networks. To take use out of these advantages we expanded our system by extracting membership functions out of neural networks which are used to transfer the knowledge into fuzzy rules.

1. Neural Unification

We have investigated an approach that is close to the problem representation. The main idea is to use Kohonen’s Self-Organizing Feature Maps (SOFM) [5] for the representation of the atoms and functors in a term. SOFM have the property that similar input data (in this case atoms and functors) are represented in a close neighborhood in the feature map (relating to their semantical context). For each atom, resp. functor, in the logical statement the input vector for the SOFM is generated as follows: each component of the feature vector represents the number of occurrences of the given atom/functor in a (sub-)term, whereby the number after the feature term refers to the arity of the term. The length of the vector is the number of possible (sub-)terms. The training of the SOFM with these input vectors results in a map, called Input Feature Map (IFM).

A special designed relaxation network, called Cube, performs the unification by determining the most common unifier. For each argument position of the unifying terms a layer of neurons is constructed, having the same topology as the IFM. For each occurrence of a variable in the given Prolog program a vector of neurons, called Variable Vector, is constructed. The encoding of the vectors is the same as for the input vector of the IFM. Cube and Variable Vectors are connected through neurons leading to and from a vector of neurons called Pairs. Each neuron in the Pairs vector encodes two argument positions that have to be unified. Lateral connections between the pairing neurons activate identical argument positions. With an simple threshold neuron operating on the Pairs neurons the occur check can be realised [15]. The activation functions of the different neurons are constructed such that if the network has reached a stable state (relaxation process), the unification process can be performed [15]. In order to actually calculate the most common unifier a special SOFM, called Output Feature Map (OFM), is constructed. Weights of this feature map are the activations of the Cube neurons. If the Variable Vector of a variable is used as input pattern to the OFM, the neuron representing an instance of that variable responds.

In our network tests like occurrence and clash are implemented such that they can be calculated in parallel and during the relaxation process. Unification is performed via a relaxation neural network. If this network has reached a stable state the most common unifier can be read out using the OFM. It can be proven that our network performs the unification process precisely [15]. Real world applications of logic programming, in particular in expert systems, require more than exact reasoning capabilities. In order to perform fuzzy unification the AND resp. OR neurons of the relaxation networks have to be modified. Instead of the AND resp. OR function in the neurons with connections from the Variable Vectors to the Pairs neurons the activation function is changed to the minimum resp. maximum of the two input activations [15]. We have tested the system with different programs consisting of a small Prolog database, simplification of algebraic terms, symbolic differentiation, and the traveling salesman problem [15].

2. Introspection

Many symbolic knowledge processing systems rely on programs that are able to perform symbolic proofs. Interpreters for the programming language Prolog are examples of such programs. The usage of Prolog interpreters for symbolic proofs, however, implies a certain proof strategy. But in case of failure of a partial goal, the interpreter backtracks systematically to the last choice made without analyzing the cause of failure. Even for simple programs, this implicit control strategy is not sufficient to obtain efficient computations. Neural networks can be used to automatically optimize symbolic proofs without the need of an explicit formulation of control knowledge [16]. We have realized an approach to learn and store control knowledge in a neural network. Input to the neural network is the Prolog clause to be proved. The output is an encoded structural description of the subgoal that is to be proved next. In order to do a comparison we have realized three different neural networks for that problem [16]: ART1 extended to supervised learning mode [2]; Backpropagation [8]; and Kohonen's Self-Organizing Feature Maps (SOFM) [5].
A meta-interpreter generates training patterns for the neural network. It encodes successful Prolog proofs. Trained with these examples of proofs the neural network generalizes a control strategy to select clauses. Another meta-interpreter, called generating meta-interpreter (GMI), is asked to prove a goal. The GMI constructs the optimal proof for the given goal, i.e. the proof with the minimal number of resolutions. The optimal proof is found by generating all possible proofs and comparing them with reference to the number of resolutions. For an optimal proof each clause-selection-situation is recorded. A clause-selection-situation is described by the features of the partial goal to be proved and the clause which is selected to solve that particular goal. The clause is described by a unique identification and two different sorts of information concerning the structure of arguments are used: the types of arguments and their possible identity. For the types of arguments a hierarchical ordering of the possible argument types is used. The encoder takes the clause-selection-situation and produces a training pattern for the neural network. The encoding preserves similarities among the types. The neural network is trained with the encoded training patterns until it is able to reproduce the choice of a clause for a partial goal. A query is passed to an optimizing meta-interpreter (OMI). For each partial goal the OMI presents the description of the partial goal as input to the neural network and obtains a candidate-clause for resolution. With this candidate the resolution is attempted. If resolution fails, the OMI uses Prolog search strategy as default.

Our system allows to generalize the earned control knowledge to new programs. In order to do this, structural similarities between the new program and the learned one are used to generate a mapping of the corresponding selection situations of different programs.

We have tested our approach using several different Prolog programs, for example programs for map coloring, travelling salesman, symbolic differentiation and a little expert system [16]. It turned out, that almost all neural networks were in principle able to learn a proof strategy. Best results in reproducing learned strategies were obtained with the modified ART1-network which reproduced the optimal number of resolutions for a known proof. For queries of the same type (same program) that were not used as training data, however, the SOFM turned out to be the best. A proof strategy using this neural network averaged slightly over the optimal number of resolutions even for completely new programs but well below of the number of resolutions a Prolog interpreter needs. The backpropagation network we used was the worst for both cases [16].

3. Summary

We showed several meaningful ways to integrate neural networks with knowledge-based systems. Concerning Neural Unification we have studied a neural unification algorithm using Self-Organizing Feature Maps by Kohonen. This neural unification algorithm is capable to do the ordinary unification with neural networks whereby important problems like the occur-check and the calculation of a most common unifier can be done in parallel. In Introspection we have tested several different neural networks for their ability to detect and learn proof strategies. A modified Self-Organizing Feature Map has been identified to yield best results concerning the reproduction of proofs made before and for generalizing to completely new programs. In the field of Structure Detection we have developed a combined toolbox to detect structures that a Self-Organized Feature Map has learned from subsymbolic raw data by the collective behaviour of assemblies of neurons. Data stemming from measurements with typically high dimensionality can be analyzed by using an apt visualization of a Self-Organizing Feature Map (U-matrix methods). The detected structures can be reformulated in the form of symbolic rules for Integrated Knowledge Acquisition by a sophisticated machine learning algorithm sig*

The usage of neural networks for integrated subsymbolic and symbolic knowledge acquisition realizes a new type of learning from examples. Unsupervised leaning neural networks are capable to extract regularities from data with the help of apt visualization technics. Due to the distributed subsymbolic representation, neural networks are, however, not able to explain their inferences. Our system avoids this disadvantage by extracting symbolic rules out of the neural network. It is possible to give an explanation of the inferences made by the neural networks. By exploiting the properties of the neural networks the system is also able to effectively handle noisy and incomplete data. Algorithms for neural unification allow an efficient realization of the central part of a symbolic knowledge processing system and may also be used for neural approximative reasoning. Introspection with neural networks frees the user and programmer of knowledge processing systems to formulate control knowledge explicitly.
4. Acknowledgement

We thank Ms. G. Guimaraes, Mr H. Li, and Mr V. Weber for the helpful discussions. We thank all the students of the University of Dortmund, Germany who worked on preliminary versions of our systems. This research has been supported in part by the German Ministry of Research and Technology, project WiNA (contract No. 413-5839-01 IN 103 C/3) and by the Bennigsen-Foerde price of NRW.

5. References