Concept Support as a Method for Programming Neural Networks with Symbolic Knowledge

Erich Prem, Markus Mackinger, Georg Dorffner
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3,
A-1010 Vienna, Austria, Email erich@ai.univie.ac.at

G.Porenta, H.Sochor
Kardiologische Universitätsklinik,
Garnisongasse 13, A-1090 Vienna, Austria

January 18, 1993

Abstract

Neural networks are usually seen as obtaining all their knowledge through training on the basis of examples. In many AI applications appropriate for neural networks, however, symbolic knowledge does exist which describes a large number of cases relatively well, or at least contributes to partial solutions. From a practical point of view it appears to be a waste of resources to give up this knowledge altogether by training a network from scratch. This paper introduces a method for inserting symbolic knowledge into a neural network–called "concept support." This method is non-intrusive in that it does not rely on immediately setting any internal variable, such as weights. Instead, knowledge is inserted through pre-training on concepts or rules believed to be essential for the task. Thus the knowledge actually accessible for the neural network remains distributed or –as it is called– subsymbolic. Results from a test application are reported which show considerable improvements in generalization.

Keywords: Neural Networks, Connectionism, Preprogramming, Application

1 Introduction

The common view of neural networks and rule-based approaches to diagnosis or classification tasks in AI is that the former always acquire their knowledge through training, while in the latter case knowledge is pre-programmed as rules. Thus, each method has its type of applications where they appear most appropriate, in the case of neural networks applications where no or little explicit knowledge exists or is known. As a result, neural networks are usually trained with random inital configurations, the only source of knowledge being the training examples. In many real world cases, however, partial symbolic knowledge on how to solve the problem exists beforehand and is already formalized. It seems to be a waste of effort not to make use of this knowledge. The following are possible reasons why finding a way of inserting explicit knowledge into a neural network can be desirable.

- Learning from scratch (*tabula rasa*) is a very costly process. When starting with randomly distributed weights, the training examples have to be presented to the network many times before any success becomes visible.
- The number of training samples needed to train an "empty" network can be very large, which helps in overcoming the problem of small training sets one is often faced with.
- Not all the necessary knowledge might actually be contained in the training samples themselves. Existing rules could be used to replace this missing part.
- Pre-programming can prevent spurious states or local minima the network might otherwise be prone to reach.
- It can be easier to interpret or debug networks into which rules have been injected.

This paper reports on results of applying a specific technique on inserting symbolic knowledge into a network, henceforth called *concept support*. One of the basic properties of concept support is that it is a method of programming neural networks in a non-intrusive fashion, i.e. without immediate alterations of their internal structure, such as weight matrices. This method and its variations—support of rules, and support of related concepts—are introduced emphasizing their advantages and value for practical applications. Concrete results from using a test application in the medical domain are described and discussed.

2 Concept Support Techniques

We call methods that try to incorporate a priori knowledge into neural networks without immediate access to the internal structure (weights) *concept support techniques*. The basic idea is to pre-train the network on the basis of the symbolic knowledge, so as to obtain a weight matrix on which further learning with the training samples is based. This way the inserted knowledge–i.e. the knowledge actually usable or accessible for the neural network–is kept at least *weakly* distributed (see van Gelder 1990), or subsymbolic. No attempt at symbolically interpreting any internal entity (unit or connection) is necessary, as would be if weights were to be preset explicitly (cf. [Yang 90, Sethi 90]). The main goal is not so much to reduce training times, but rather to improve the generalization capabilites of the network.

Several variations on concept support can be distinguished:

1. the pretraining on the basis of rules describing a subset of cases of the desired input-output mapping. The network for pre-training in this case is the same as for the actual task and is trained in two phases; first on one or more rules, secondly on the training samples. Thus the rules complement the training samples in supplying the knowledge to the network.

2. the pretraining on the basis of symbolic concepts believed to be relevant for solving the task. This idea originated in the work by Wiklicky (1989) and gives the method its name–*concepts* relevant to the task are *supported* during training. In this case, the output layer of the network

is different during the two phases. During pretraining the output represents the relevant concepts, during regular training it represents the desired output.

3. the pretraining with concepts as in (2), but with a smaller hidden layer, while at regular training additional hidden architecture is used. This is the original method introduced in Wiklicky (1989). This way, during regular training the inserted knowledge is confined to only a part of the network, leaving the rest more freedom to self-organize.

An important aspect for all methods of concept support–or all methods for inserting knowledge, for that matter–is that the knowledge supplied during pre-training need not be totally consistent, complete in any sense, or even correct in order to ensure usable results. In other words, "overwriting" or modifications of inserted knowledge should always be possible due to training with the application data. The worst case should be that training time is increased through misleading or incorrect knowledge. As concept support techniques do not affect the basic process of the neural networks, such a property comes almost automatically.

We have tested methods (1) and (2) on a specific application described below. Both methods have been proven to be useful. Especially in the case of supporting rules, considerable improvements of the generalization behavior could be observed. Before actually describing the application we take a brief look at related work.

3 Related Work

The importance of overcoming technical problems of neural networks which are related to nonsymbolic representation, a priori rules and concepts have been pointed out by several authors, e.g. [Gallant 88, Hendler 89]. While many of the proposed methods try to combine neural networks with symbolic methods there is not so much work on incorporating knowledge *and* keeping the nature of representation in the model distributed.

Methods which have shown to achieve both are described in Suddarth (1990) or Yeoung-Hu (1990). In both cases there is an additional output

information during training that helps to guide the search for a solution to the training problem. Both methods not only use the desired output pattern during learning but expand the training pattern in order to train the network on a different but related task. One part of the training pattern represents the original task, the other one additional information about the input pattern. One difference to the method we describe below is that both parts of the output (original task and additional information) are presented within one single pattern. Moreover, our method always separates a concept support phase and a phase of training the network on the desired task. This has the advantage that initalization of a priori knowledge and later refinement can be performed separetely, which is useful for systematic engineering or security reasons.

The idea of creating a specific hidden unit representation which supports the generalization capability of the network has e.g. been proposed in [Lendaris 90]. A more redundant encoding scheme (compared to a straightforward one) is used as a representation in a hidden layer of the network. This representation makes it easier for the net to correctly learn the desired generalization, but leads to a local representation in one hidden layer. As opposed to our method, this technique can be regarded as actually combining two different networks connected by the designed hidden layer representation.

Solutions to the problem of creating networks that are specialists in a specific domain and methods of constructing *domains of expertise* and hierarchies are principally discussed in [Jacobs 88], but no practical methods or examples are mentioned there. Jacobs proposes to train the net on a series T_i of tasks which–very informally–*converge* to the desired task T_n. Our proposal of designing these tasks by means of existing rules (the *a priori* knowledge) is presented in the next sections.

4 An Example in the Medical Domain

Computer based image interpretation of thallium-201 scintigrams served as a testbed for incorporating rules into neural networks. The task to be performed by the neural network is the asessment of coronary artery disease (CAD) using the parallel distributed approach. Earlier endeavours to

solve this problem with a neural network suffered from relatively low prediction rates and the lack of possibilities to incorporate *a priori* knowledge [Porenta 88]. Normally, the medical expert interprets the scintigram picture; he decides on the likelihood of CAD, number of affected vessels and location of affection. Results given by the expert can be verified by comparison with angiography. Thus two reference methods exist which can be chosen to generate the target patterns for the network.

The scintigraphical images of the heart-scan are converted into 45 numeric values representing scans under stress, redistribution, and the washout, which are input to the network. The network used for this study therefore consists of 45 input, 23 hidden, and 1 or 2 output units - depending on the task. It was trained by using error backpropagation.

A priori knowledge about the task consists of several heuristic symbolic rules. These rules either describe criteria for a patient to be classified as ill or for identifying the affected region of the heart and were obtained by consulting a medical expert. It must be stressed that all of these rules are very heuristic, i.e. cover only a limited percentage of the input space. Reasons therefore are inaccuracies in the input data, enourmous physiological differences between patients and irregularities during recording patient data.

4.1 The Method of Rule Support

The method of supporting the network with the rule as mentioned above consisted of the following. In the first phase (which we call the "preprogramming phase") the network was trained on the rule alone. For this, the heart scans of the training samples were used as input, while the diagnosis (pathological or not) as calculated *by the rule* applied to this input was used as target for the generalized delta rule. After training the following values were computed for comparsion.

- the percentage of cases of the whole data set that the *rule* classified correctly (as compared to the reference method)—value a.
- the percentage of cases for which the network trained this way correctly predicted the outcome of the rule—value b. This is some kind of

indication how well the network implemented the rule.

- the percentage of cases of the whole data set the *network* classified correctly, as compared to the reference method—value c.

After that a second training phase was added (which we call the "refinement phase"), in which the preprogrammed network was further trained on the regular training samples (i.e. the previous set of heart scans as input, and the diagnosis by the reference method as target). Again, the percentage of cases (in the test set) which the refined network classified correctly (compared to the reference method) was calculated (value d).

Learning was continued until all cases of the training set were classified correctly, while using a threshold of 0.5 for decision (i.e. everything above that threshold was interpreted as 1, everything below as 0). To calculate the said prediction rates (values a through d) the same threshold of (0.5) was used.

4.2 Diagnosing the Location of the CAD

To make the diagnosis more specific it is possible to distinguish between the left and the right region of the heart. This distinction of affected vessels is a more difficult task even for the medical expert. This is why angiography was used as reference method for this task. The network used two output units for the right/left distinction of the physiological region. 23 examples were used as a training set, 82 as the test set. Training the net on randomally selected examples achieved prediction rates of 44 % after 50 epochs.

The network was trained using the following symbolic rule with respect to physiological areas for the left/right-distinction.

> *If one scan value is below 60 or the washout rate is negative then classify the case as pathological.*

A set of numeric values referring to the corresponding position of a vessel in the picture was tested on having one value below 60 or representing a negative washout rate. In Fig.1 prediction rates of the symbolic rule are compared with the normally trained net, the network trained on the rule and with the

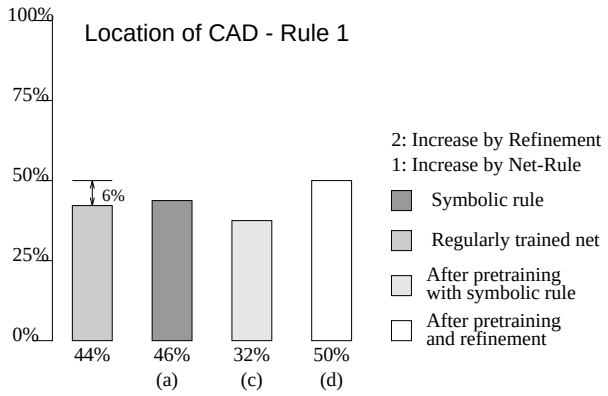

Figure 1: Results of supporting rule 1 in a network trained to predict the location of disease. The prediction rate of a regularly trained network is compared to the rule itself (value a), the network trained by the rule (c), and the refined network (d). For a description of the values see text.

additionally refined network. The difference in the prediction rate between the symbolic rule alone (value a) and the network trained on the rule (value c) is 29 %. Additional training with the original task increased the prediction rate by another 4 % (value d). The rule itself was learned to 55 % (value b), i.e. the network correctly implemented the rule with this percentage.

4.3 Support with a Relevant Concept

So far we have described the support of symbolic *rules*. As mentioned earlier the support of *concepts* relevant for the given domain can also be used to improve generalization. One idea about the heart-scan data is that there are differences in the scans of men and women. Thus, a patient's sex was decided to be a relevant concept with which the network should be supplied. Again, two training phases were conducted. In this experiment the task for the network consisted in predicting the disease without locating it. In the first phase a net with two output units was trained on the male/female distinction. In the second phase the output units were replaced by a single unit and the resulting network was trained on the original task. Fig. 2 shows the results. When taking the original threshold of 0.5, the percentage of correctly classified cases rose from 60 % to 73 % due to preprogramming with the concept.

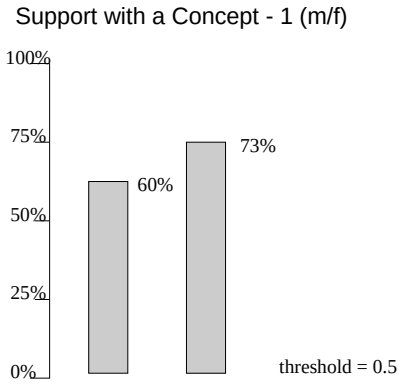

Figure 2: Results from supporting the male/female-distinction as a relevant concept. The prediction rate of a network that was pretrained on this distinction is compared to a regularly trained network. The results are shown for two different output thresholds, i.e. the criterion to interpret the output as a binary value.

5 Discussion

The results show that applying the method of concept support consistently improves the generalization performance of a multi-layer feedfoward network trained on a classification task, compared to reference tasks with the same conditions but without support. It provides a feasible way of incorporating existing symbolic knowledge about the task domain into the network learning. Very often the task for an application of this kind is to obtain a neural network that generalizes as well as possible to novel cases, in order to make use of it in practice (in clinical routines, in our case). Introducing knowledge in the described way therefore proves a considerable advance toward achieving this task.

The two main effects of the method are that

- no alterations to the network architecture, or interpretation thereof, is necessary for achieving the results.

- the generalization increases after regular training on the data samples. This phase can therefore indeed be seen as a refinement of what was given by the rule. In this sense, rule and distributed network complement each other optimally.

In the following we attempt to give a qualitative explanation of the observed effects. A rule, by being symbolic and formulated by a human expert, covers some of the "obvious[1]" features in the input that lead to a certain diagnosis. At the same time the rule is *reductionist* in that it concentrates on a perspicuous part of the input. A network only trained on samples cannot easily profit from such obvious dependencies but has to extract them from the presented patterns in a tedious and costly process. This is how the rule can help the network by "telling" it what features in the input to turn to.

In our particular case, during the preprogramming phase the network has to learn to classify as pathological only those cases that fulfill the obvious features in the input (such as "one value below 60"), therefore becoming sensitive to them. At the same time, however, the network always works in a *holistic* way in that it always considers all inputs during learning. So it is capable of discovering correlations among the not-so-obvious features in the input that also lead to the same diagnosis even in the absence of the obvious feature. The rule might also supply information about cases that might not be contained in the training set. The refinement phase then adds cases which are not covered by the rule or its fuzzy neighborhood, leading to a further increase (value d).

This property of a rule does not seem specific to this particular application but rather appears to be more generally the case for many problems which lend themselves to neural network solutions. If knowledge exists for such problems it mostly consists in a handful of heuristic rules (otherwise a rule-based approach would appear more feasible). Those rules can help the neural network approach in the described sense. In this way, even a single rule or a few rules can be of use. It was not even realistic to expect much more knowledge than that described in the previous section. Therefore, although it cannot formally be proven at this point, the approach appears to

[1]I.e. "easily" identifiable for the expert

be viable for a great variety of practical applications. It is assumed that the method extends to larger sets of rules also. At least it can be proven that the method can never prevent the network from learning or make the overall generalization rate worse.

The following additional interesting observations can be made

- The training time for the combined training (pre-training with rules or concepts plus regular training with the sample data) was comparable to the training time without concept support. The important result is that it is not increased considerably.
- In any case, training times to improve results in the second phase tend to be short. In other words, refinement of the performance achieved through learning the rule to higher levels based on the real-world examples is rather quick.
- The rules or concepts used during the support phase need not be checked on their consistence, relevance, or completeness. Rules, even only crude ones, or concepts that are in some sense relevant lead to at least some improvement. Totally irrelevant rules at worst increase training time, but usually cannot disturb the final results (in our case, it was even improved slightly, probably for reasons of random matching of the "nonsense rule"). As a result, any piece of knowledge, including heuristics, can be helpful in improving the results.
- The usual search for an optimal training set, which usually plagues an application with the above-mentioned goals (a good generalizer), can be reduced considerably. The results have shown that concept support can refine the performance even for training sets that are less representative as needed for learning from scratch.

6 Conclusion

This paper has introduced a method for incorporating knowledge into a neural network during the coarse of training. By training the network on the symbolic realization of the knowledge (rules or single concepts) this knowledge is transformed into a distributed version, of which the network can

make optimal use. The results of several experiments in a medical diagnosis task have shown that the method consistently improves generalization performance considerably above the levels reached by crude training from scratch. Thus it has proven as a viable and useful method for combining symbolic knowledge with subsymbolic techniques, which is an important general goal in recent artificial intelligence endeavors.

7 Acknowledgements

This research is part of the ESPRIT-II project NEUFODI (Neural Networks for Forecasting and Diagnosis Application). Neufodi is sponsored by the EC commission and the Austrian Industrial Research Promotion Fund as ESPRIT-II project No.5433 and is conducted in cooperation with the Babbage Institute for Knowledge and Information Technology (Belgium); Lyonnaise des Eaux Dumez (France); Elorduy, Sancho y CIA, S.A.; Laboratorios de Ensayos e Investigaciones Industriales (both Spain).

References

[Gallant 88] Stephen I. Gallant 1988. *Connectionist Expert Systems.* Comm. of the ACM, Vol. 31(2):152-169, Feb. 1988.

[vanGelder 90] Tim van Gelder 1990. *Why Distributed Representation is Inherently Non-Symbolic.* In: Georg Dorffner (Ed.), Konnektionismus in Artificial Intelligence und Kognitionsforschung, Proc.of the 6th Austrian AI-Conference, Springer Berlin 1990, pp. 58-66.

[Hendler 89] J.A.Hendler 1989. *On the Need for Hybrid Systems.* Connection Science 3(1), pp.227-229.

[Jacobs 88] Robert A.Jacobs 1988. *Initial Experiments On Constructing Domains of Expertise and Hierachiers in Connectionist Systems.* In: D.S.Touretky (Ed.) Proc.of the Connectionist Models Summer School 1988, Morgan Kaufman, San Mateo, CA, pp.144-153.

[Lendaris 90] George C.Lendaris, Ihab A.Harb 1990. *Improved Generalization in ANN's via Use of Conceptual Graphs: A Character Recognition Task as an Example Case.* In: Proc.IJCNN 1990, San Diego, CA, 1990, pp.I 551-555.

[Porenta 88] Gerold Porenta, G.Dorffner, J.Schedlmayer, H.Sochor 1988. *Parallel Distributed Processing as a Decision Support Approach in the Analysis of Thallium-201 Scintigrams.* In: Proc. Computers in Cardiology 1988, Washington D.C., IEEE.

[Sethi 90] I.K.Sethi 1990. *Entropy Nets: from Decision Trees to Neural Networks.* Proc.of the IEEE, vol.78, 10, Oct.90, pp 1605-1613.

[Suddarth 90] S.C. Suddarth, Y.L. Kergosien 1990. *Rule-injection hints as a means of improving network performance and learning time.* In: L.B. Almeida, C.J. Wellekens (Eds.), Neural Networks, Proc. of the 1990 EURASIP Workshop, Springer, Berlin, 1990.

[Yang 90] Qing Yang, Vijay K.Bhargava 1990. *Building Expert Systems by a Modified Perceptron Network with Rule-Transfer Algorithms.* Proc.of the IJCNN 1990, San Diego, CA, pp.II 77-82.

[Yeoung-Ho 90] Yu Yeoung-Ho, Robert F.Simmons 1990. *Extra Output Biased Learning.* Proc.of the 1990 IJCNN, San Diego, CA, 1990, pp.III 161-164.

[Wiklicky 90] Herbert Wiklicky 1990. *Neurale Strukturen: Möglichkeiten der Initialisierung und symbolisches Wissen.* Thesis, University of Vienna, 1990.