NEULA: A hybrid neural-symbolic expert system shell

Patrik Floréen, Petri Myllymäki, Pekka Orponen, Henry Tirri

University of Helsinki, Department of Computer Science
Teollisuuskatu 23, 00510 Helsinki, Finland*

Abstract

Current expert systems cannot properly handle imprecise and incomplete information. On the other hand, neural networks can perform pattern recognition operations even in noisy environments. Against this background, we have implemented a neural expert system shell NEULA, whose computational mechanism processes imprecisely or incompletely given information by means of approximate probabilistic reasoning.

1 Background

Most current artificial intelligence systems are incapable of handling imprecise and incomplete information. This is a serious drawback, as in most cases it is a totally hopeless task for a programmer to capture all knowledge of the environment. Much research effort has been expended to improve the robustness of artificial intelligence programs, but very little progress has been made.

On the other hand, current neural network research indicates that it is possible to perform pattern recognition operations, such as categorization, even in very noisy environments. We have therefore investigated the interesting question of whether the robustness problem of traditional artificial intelligence applications can be attacked by neurally inspired techniques for knowledge representation.

Inspired by the general idea of *hybrid neural-symbolic* systems (e.g. [4]), we have introduced in the NEULOG[1] project a knowledge representation scheme for robust storing of hierarchically related concepts. The scheme is based on a neural representation structure, which bears resemblance to a semantic network. However, instead of the traditional artificial intelligence interpretation of the network as a purely syntactic variant of first-order predicate calculus, the network performs computations implementing a Bayesian reasoning model, which is capable of inference in the presence of incomplete, imprecise and even inconsistent information.

*Email: Firstname.Lastname@Helsinki.FI

[1]The NEULOG project was supported by the Finnish Technology Development Center (TEKES) under its FINSOFT program, and by the Academy of Finland.

2 Representing probabilistic knowledge in the NEULA language

The prototype programming environment consists of a high-level knowledge representation language NEULA (NEUral LAnguage), a compiler that is capable of realizing the symbolic syntax as a neural network, and a neural network simulator. The NEULA system works as an expert system shell. There are two types of users: expert users who provide the knowledge, and end users who make queries against that knowledge. The expert users write NEULA programs. A NEULA program consists of a hierarchically organized set of concept descriptions. As a simple example, consider the following description of the inhabitants of a zoo:

```
concept animal (100) is basic with
    offspring : [ living (20), eggs (80) ];
    can       : { swim (70), fly (29), walk (49) };
concept mammal (20) is animal with
    offspring : [ living (20) ];
    can       : { swim (10), fly (1), walk (19) };
concept bird (30) is animal with
    offspring : [ eggs (30) ];
    can       : { swim (10), fly (29), walk (30) };
concept fish (50) is animal with
    offspring : [ eggs (50) ];
    can       : { swim (50), fly (0), walk (0) };
concept dolphin (2) is mammal with
    can       : { swim (2), walk (0) };
concept penguin (6) is bird with
    can       : { swim (6), fly (0), walk (6) }.
```

Here, a description of a concept consists of a reference to its immediate ancestor (if any), together with a list of attributes and their value distribution for objects belonging to this conceptual class. There are two types of attributes: exclusive (indicated by the square brackets "[ ]" enclosing the list of possible values) and multi-valued (indicated by the curly braces "{ }"). Exclusive means here that any individual realization of a concept has to possess exactly one value of the given attribute: for example, the offspring of any particular animal in the zoo has to be either living or eggs.

The parenthesized numbers indicate the "frequency" of a given value for an attribute, or at the header of a concept declaration, the "frequency" of all objects in that concept class. For example, the sentence "**concept** animal (100) **is basic**" defines the total number of all the animals in the zoo to be 100. This may be the actual number of the animals, or just a relative proportion (i.e., denoting 100% with 100). Similarly, the definition "**concept** mammal (20) **is** animal" states that 20% (20/100) of the animals are mammals. Furthermore, in the definition "offspring : [living (20)];" it is stated that 100% (20/20) of the mammals give birth to living offspring (0% lay eggs), and from the definition "can:{ swim (10), fly (1), walk (19) }", it is concluded that 50% (10/20) of the mammals can swim, 5% (1/20) of them can fly, and 95% (19/20) can walk. Hence, we have implicitly defined probabilistic rules which state that, for instance, the probability of a randomly picked animal in the zoo being a mammal is 0.20, and on the other hand, if we randomly choose one mammal among all the mammals, the probability of this mammal being able to walk is 0.95. We could of course easily change the syntax to have the actual probabilities in the code instead of the relative numbers, but we have found the current indirect way to express the probabilities more user friendly, because it is easier for people to think in terms of numbers of instances than to estimate relative frequencies.

3 Compiling the NEULA code

The probabilistic rules given in the NEULA program are transformed to a neural network using the NEULA compiler. The compiler first creates a description of the code as a *belief network* representation [22], which consists of a set of random variables and probabilistic dependencies between these variables. In our scheme, each concept and (attribute:value) pair in the NEULA code is regarded as one variable in the belief network. The conditional probabilities attached to the dependencies are calculated from the numbers given in the code. The belief network is then transformed to a harmonium-type neural network [25]. The interconnection weights in the neural network are calculated from the conditional probabilities in the belief network. For each variable in the belief network, there will be a corresponding node in the neural network: these are called *visible units*. There is also a large number of other nodes, which have no such direct interpretation; these *hidden units* are invisible to the user. In general, the number of hidden units could be exponential in the size of the belief network, which would make practical applications of this system impossible. However, the fact that a NEULA description consists of a *hierarchy* of concepts enables us to reduce the number of hidden units to grow just linearly in the size of the description (see e.g. [21]).

4 Query processing

The neural network created from the compilation can be seen as a model of the world described in the NEULA code. Each visible node, corresponding to an (attribute:value) or (class:instance) pair in the actual world described, has an activity value, which represents the likelihood of the corresponding fact being true in the present world. Changing the activity values ("truth values") of the nodes corresponds to changing our expectations of the facts in our world. The end user can make queries against the knowledge encoded in the neural network. The user states a query to the network by permanently setting ("clamping") the activity values of a selected set of nodes. In the NEULA system, this can be accomplished by using a graphical point-and-click interface called *NeulaTool* [20]. For example, the question "Which animal is in question if we know that it cannot walk and it has living offspring?" can be performed by clamping the value of the node (offspring:living) to 1.0 — which implicitly clamps the node (offspring:eggs) to 0.0 — and the value of the node (can:swim) to 0.0 (see the window titled "Clamped" in figure 1).

The network processes the query by performing *iterative computations* which change the activity values of the unclamped nodes in the neural network, according to the given model of the world. After the computation, the answer to the query may be retrieved by examining the activity values of the visible nodes, i.e. the truth values of the corresponding variables (see the window titled "Focused" in figure 1). It is important to notice here that the user does not have to specify all the attributes of the corresponding animal, in this case the dolphin, to get the correct answer. What is more, some of the given descriptors might even be incorrect or inconsistent with each other or with the original description.

During query processing, the system is going through a large number of different worlds, trying to find the best combination of nodes and their activity values consistent with the clamped variables, according to some goodness measure. In an abstract sense, the computation scheme can be seen as a search algorithm. However, this search procedure is very complicated as all the nodes affect the result of the computation during the iterative process. On the other hand, the result of the computation is a complete model of the world with truth values for *all* the variables. This makes it possible to study correlations between different variables: for example, it is possible to get an answer to the ques-

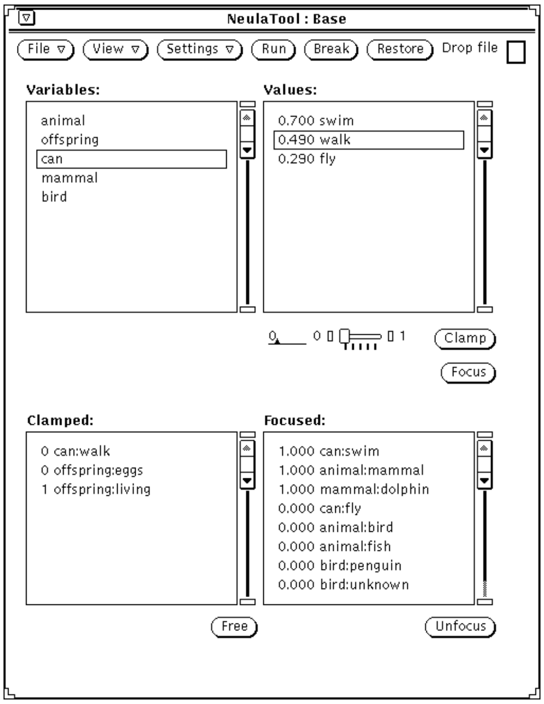

FIGURE 1. Base window of the NeulaTool program after processing a query.

tion "what can an animal which cannot swim but has living offspring do?" by studying the activity values of the nodes (can:swim) and (can:fly) in the Focus window.

In addition to the concept recognition queries, it is possible to perform questions where the concept is given, and the properties are asked. For example, the question "Assuming that we are dealing with a dolphin, what can this kind of animal do?" is done by clamping the node (animal:dolphin) to 1.0, and studying the values of the nodes (can:*). Although not all the properties of the dolphins are described in the NEULA code (e.g., whether they can fly or not), the system is able to provide an answer. The two types of queries may also be arbitrarily mixed with each other, allowing very powerful forms of queries. For example, sometimes it could be useful to be able to ask: "Assuming that we are dealing with a dolphin that *can* walk, what can this kind of animal do?". In some sense, answering a query resembles an *analogical reasoning* process: the system first identifies the object that is most consistent with the given input, and then provides an answer using the properties of this object.

5 The computational model for probabilistic reasoning

In the previous section, we described the queries that can be performed in the network. But we still need some kind of "goodness of fit" measure for the different states of the network, some kind of a mathematical model for the calculations. In earlier versions of the system [6, 7], we used a heuristic computational scheme. This scheme was not satisfactory, however, as we wanted to have an exact interpretation of the truth values given to the user. This can be done: neural network models are, due to their ability to use massive parallelism in the computations, an ideal tool for implementing the so called Monte-Carlo family of Bayesian reasoning methods, especially a method called Gibbs sampling. We have developed two different techniques for implementing efficient approximations of this kind of reasoning by our neural network models. The methods are based on the theory of *Markov random fields* [8, 13], and their close relationships to Bayesian networks, as discussed in [12, 14, 15, 26]. The techniques developed provide a sound basis for the probabilistic interpretation of the truth values given as the result of a query.

In both our models, all the nodes are actually binary, being at any iteration step either in state 1 (representing the value "true"), or 0 ("false"). In the first technique, described in [21], we are able to approximate the probability of each of the variables by the *frequency*

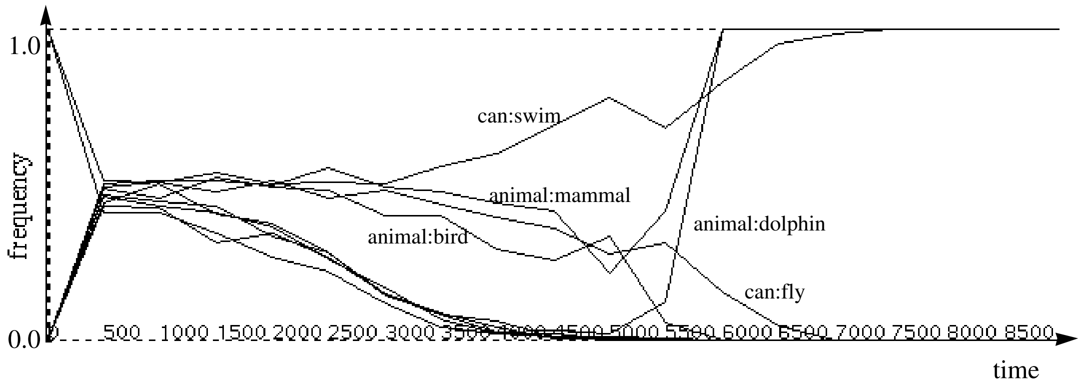

FIGURE 2. Node activity values during the "dolphin query" processing.

of the corresponding node having been in the state 1, resulting in an estimate of the conditional Bayesian probabilities for each of the objects and (attribute:value) pairs, given the current input. According to Pearl [22] and Geman&Geman [8], the estimates given by the network converge to the correct probability as the simulation continues. Consequently, the accuracy of the answer becomes better and better with increased running time. When no significant changes occur any more, it may be assumed that the result is very close to the exact answer, and the computation may be stopped. The main problem with this approach is the fact that it is very difficult to *a priori* find out how long time this kind of convergence will take. On the other hand, instead of simulating the neural network model in a conventional computer, the computation may be performed in a special parallel environment (neural net emulator or chip), which enables enormous speed-ups in performance.

The second model, described in [17], uses the *simulated annealing* scheme (see e.g. [1]). The simulated annealing technique forces the network to eventually settle down to the "maximum entropy" state, which is the most probable state of the network consistent with the given input query. A typical plot of the state 1 occurrence frequencies of the nodes during the "dolphin query" processing is shown in figure 2.

The neural network model used in the second model is the *harmony network* [25], which is a kind of a generalization of the Boltzmann machine model [10, 11]. In this scheme, the user is given as output only either of the judgements "true" or "false" for each of the variables. This kind of behaviour is very useful in environments where the system is expected to make a definite decision, even under uncertainty.

6 Conclusions and future development

Although the NEULA system uses a neural network as the implementational platform, this representation is completely hidden from the user. Hence, from the user's point of view, NEULA is just a sophisticated expert system shell, which is able to process complicated types of queries. The main advantage of the NEULA system over traditional uncertainty management in expert systems is in its strong theoretical background: the truth values given to the user are approximations of actual Bayesian probabilities, not heuristical certainty factors, as in many current expert systems. The Bayesian framework is an attractive theoretical model for reasoning with uncertainties, but so far it has had one major drawback: in general, it seems to be impossible to calculate the conditional probabilities exactly in reasonable time [5], so in practice some kind of approximation is needed. The new sampling methods developed offer powerful tools for this kind of approximation. As neural nets allow parallelization of such models, they are a natural choice as the implementation platform.

Compared to other neural network packages, the NEULA system offers the user an opportunity to be able to specify the structure of the network and initial interconnection weights explicitly, without recourse to a time-consuming learning process. If needed, the interconnection weights may then be fine-tuned with a learning algorithm, although this feature is not implemented in the current version of the NEULA system. It is important to notice that the expert is not required to give *all* the probabilistic dependencies of the framework – only the very simple, and natural dependencies between an object and its attributes.

The neural network model behind the NEULA system is structurally simple and uniform. This is a clear advantage over many other hybrid knowledge representation schemes (see e.g. [4, 9, 24]), which require fairly complicated computing elements and control regimes. Consequently, the NEULA system would be much easier to implement in a real neural system. The mathematical models behind other hybrid systems are also usually more or less heuristic by nature, in contrast to the sound Bayesian interpretation of the NEULA computational model.

Section 4 demonstrates how our system can be used as a stand-alone expert system. However, after the NEULA code has been compiled, it is also possible to extract the generated network module, and integrate it with other software. As an example, we have implemented a system [18], where the NEULA module is integrated with a Prolog inference engine. This kind of system allows the user to mix probabilistic and logical reasoning.

The current NEULA system is able to handle only discrete values of the attributes. Although this form of knowledge is common in many application areas, the current NEULA system cannot be used in, e.g., low-level image or speech processing, which are dealing with raw numerical data. Therefore we are also studying methods for linking NEULA with a front-end module that transforms numerical data into discrete values, which can then be fed to the NEULA system as input. One possible technique for this kind of data discretization was studied in cooperation with AT&T Bell Laboratories by implementing a program for object orientation detection [16]. Another approach using instance-based reasoning techniques is described in [19].

The main problems with the current system concern its efficiency: simulated annealing techniques require a lot of iterative processing, and consequently their implementations on conventional serial computers can be excruciatingly slow. A parallel implementation of the NEULA simulator is currently under construction for the 100-transputer Hathi-2 system at Åbo Akademi [3]. On the algorithmic front, we are also studying more sophisticated annealing techniques, such as self-adjusting annealing schedules [2] and the mean field annealing algorithm [23].

References

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* John Wiley & Sons, Chichester, 1989.

[2] J. Alspector, T. Zeppenfeld, S. Luna, A volatility measure for annealing in feedback neural networks. *Neural Computation* 4 (1992), 191–195.

[3] M.Aspnäs, R.J.R.Back, T.-E. Malén, The Hathi-2 Multiprocessor System. Report A-80-1989, Departments of Computer Science and Mathematics, Åbo Akademi.

[4] J. A. Barnden and J. B. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory, Vol. I: High Level Neural Models.* Ablex Publ. Co., Norwood, NJ, 1991.

[5] G. F. Cooper, Probabilistic Inference Using Belief Networks is NP-hard. Report KSL-87-27. Knowledge Systems Laboratory, Stanford University.

[6] P.Floréen, P.Myllymäki, P.Orponen, H.Tirri, Neural representation of concepts for robust inference. *Proceedings of the International Symposium Computational Intelligence II* (Milano, Italy, September 1989, F. Gardin and G. Mauri, eds.), 89-98. Elsevier Science Publishers, Amsterdam, 1989.

[7] P.Floréen, P.Myllymäki, P.Orponen, H.Tirri, Compiling object declarations into connectionist networks. *AI Communications* 3 (1990), 172–183.

[8] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

[9] G. E. Hinton (ed.), Special issue on connectionist symbol processing. *Artificial Intelligence* 46 (1990): 1–2.

[10] G. E. Hinton and T. J. Sejnowski, Optimal perceptual inference. *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Washington DC, June 1983)*, 448–453. IEEE, New York, NY, 1983.

[11] G. E. Hinton and T. J. Sejnowski, Learning and relearning in Boltzmann machines. *Parallel Distributed Processing* (D. E. Rumelhart and J. L. McClelland, eds.). Vol. I, 282–317. The MIT Press, Cambridge, MA, 1986.

[12] T.Hrycej, Gibbs sampling in Bayesian networks. *Artificial Intelligence* 46 (1990), 351–363.

[13] R. Kinderman and J. L. Snell, *Markov Random Fields and Their Applications.* American Mathematical Society, Providence, RI, 1980.

[14] K. B. Laskey, Adapting neural learning to Bayesian networks. *Int. J. of Approximate Reasoning* 4 (1990), 261–282.

[15] S.L.Lauritzen and D.J.Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Stat. Soc., Ser. B* 1989. Reprinted as pp. 415–448 in: *Readings in Uncertain Reasoning* (G. Shafer and J. Pearl, eds.). Morgan Kaufmann, San Mateo, 1990.

[16] R. J. T. Morris, L. D. Rubin, H. Tirri, Neural network techniques for object

orientation detection: solution by optimal feedforward network and learning vector quantization approaches. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (1990) 11 (November), 1107–1115.

[17] P.Myllymäki and P.Orponen, Programming the Harmonium. *Proc. of the International Joint Conf. on Neural Networks (Singapore, Nov. 1991), Vol. 1*, 671–677. IEEE, New York, NY, 1991.

[18] P. Myllymäki, P. Orponen, T. Silander, Integrating symbolic reasoning with neurally represented background knowledge. *Proc. of STeP-92, the Finnish Artificial Intelligence Conference (Otaniemi, Finland, June 1992)*, Vol. II, 231–240. Finnish Artificial Intelligence Society, Helsinki, 1992. Also: Workshop notes of the *AAAI-92 workshop on Integrating Neural and Symbolic Processes*, 168–172.

[19] P. Myllymäki and H. Tirri, Bayesian case-based reasoning with neural networks. Manuscript submitted for publication (Oct. 1992), 7 pp.

[20] M.Oksalahti, T.Pasonen, T.Saarikivi, M.Seppänen, M.Silvonen, NeulaTool–Neuraaliverkkojen ohjelmointiympäristön käyttöliittymä. Report C-1992-33, University of Helsinki, Department of Computer Science (in Finnish).

[21] P. Orponen, P. Floréen, P. Myllymäki, H. Tirri, A neural implementation of conceptual hierarchies with Bayesian reasoning. *Proc. of the International Joint Conf. on Neural Networks* (San Diego, CA, June 1990), Vol. I, 297–303. IEEE, New York, NY, 1990.

[22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[23] C. Peterson, J. R. Anderson, A mean field theory learning algorithm for neural networks. *Complex Systems* 1 (1987), 995–1019.

[24] L. Shastri, *Semantic Networks: An Evidential Formalization and Its Connectionist Realization*. Pitman, London, 1988.

[25] P. Smolensky, Information processing in dynamical systems: Foundations of Harmony Theory. *Parallel Distributed Processing* (D. E. Rumelhart and J. L. McClelland, eds.). Vol. I, 194–281. The MIT Press, Cambridge, MA, 1986.

[26] D. J. Spiegelhalter, Probabilistic reasoning in predictive expert systems. *Uncertainty in Artificial Intelligence* (L. N. Kanal and J. F. Lemmer, eds.). 47–67. Elsevier–North-Holland, Amsterdam, 1986.