International Journal of Neural Systems, Vol. 4, No. 3 (September, 1993) 257–267

A FLEXIBLE AND FAULT TOLERANT QUERY-REPLY SYSTEM BASED ON A BAYESIAN NEURAL NETWORK

ANDERS HOLST* and ANDERS LANSNER†
SANS — Studies of Artificial Neural Systems
Department of Numerical Analysis and Computing Science, Royal Institute of Technology
S-100 44 Stockholm, Sweden

Received 8 January 1993
Revised 29 July 1993
Accepted 17 August 1993

A query-reply system based on a Bayesian neural network is described. Strategies for generating questions which make the system both efficient and highly fault tolerant are presented. This involves having one phase of question generation intended to quickly reach a hypothesis followed by a phase where verification of the hypothesis is attempted. In addition, both phases have strategies for detecting and removing inconsistencies in the replies from the user. Also described is an explanatory mechanism which gives information related to why a certain hypotheses is reached or question asked. Specific examples of the systems behavior as well as the results of a statistical evaluation are presented.

1. Introduction

In many applications, we need to classify something we have observed or make a diagnosis from some symptoms, e.g. to identify a disease from blood tests, a car engine fault from its symptoms, or a plant species from its appearance (as can for example be done with a search key for plants). Various methods for automatic diagnosis and classification have been tried out over the years. Perhaps the most dominant approach has been rule-based expert systems which use if-then rules, supplied by the designer, to make diagnoses.[1] There are also ways to generate such rules automatically as in inductive systems.[2] Another approach is to use artificial neural networks[3–6] sometimes in combination with a rule-based system.[7] The reason to use neural networks for these tasks is that they have some nice properties such as learning capability and fault tolerance.

Here such a classification system based on a neural network is described. Instead of taking all evidence at once and finding the most likely hypothesis from that, it is designed as a query-reply system, i.e. given some information not sufficient for establishing a diagnosis, the system asks the user about features one at a time and generates a hypothesis or bases a classification on this information. This stepwise procedure has some advantages. Often it might not be clear from the start what to look for or it may be time-consuming or in other ways costly to make the necessary measurements or tests. If the system asks questions, only the evidence necessary to reach a conclusion needs to be given. A main goal here has been to provide this diagnosis system with an efficient and robust question generation capability.

The system is also augmented with an explanatory mechanism which from the state of the network finds the strongest evidence for or against an attribute in the network. This can be used to generate simple explanations as to why a certain hypothesis is reached (or not reached) or why a certain question is asked by the system. To account for the conclusion reached is an important component in an interactive classification system.

To set up the system, the network is trained with examples from some domain. Thereafter the question generation is started. It consists of a set of simple algorithms for deciding from the state of the network the "best" question in the current context. The questions are all of the form: "Does the object have the feature x?" The answers (real numbers between 0 and 1) are fed directly into the network.

* Email: aho@sans.kth.se
† Email: ala@sans.kth.se

2. The Bayesian Neural Network Model

The neural network model used here is a one-layer fully-connected Bayesian neural network.[8] In this network the activities of units are related to probabilities of stochastic events. The model is based on Bayes rule for conditioned probabilities. According to this rule the probability of an event q (e.g. a class) given a set of independent events $A = \{x_i, x_j, x_k, \ldots\}$ (e.g. features), can be calculated as

$$P(q|x_i, x_j, x_k, \ldots) = P(q) \cdot \frac{P(x_i, x_j, x_k, \ldots|q)}{P(x_i, x_j, x_k, \ldots)}$$

$$= P(q) \cdot \prod_{x_i \in A} \frac{P(x_i|q)}{P(x_i)} . \quad (1)$$

By taking the logarithm of (1), it may be rewritten as a sum[9]

$$\log(P(q|x_i, x_j, x_k, \ldots))$$

$$= \log(P(q)) + \sum_i \log\left(\frac{P(q,i)}{P(q)P(i)}\right) o_i , \quad (2)$$

where $o_i = 1$ if $x_i \in A$ and 0 otherwise.

Let us now identify events (both evidence x_i and classes q) with units in a neural network that sum their inputs according to

$$s_q = \beta_q + \sum_i w_{qi} o_i , \quad (3)$$

where s_q is the support for unit q, o_i is the output from unit i and the biases β_q and weights w_{qi} are chosen as

$$\beta_q = \log(P(q)) \quad (4)$$

$$w_{qi} = \log\left(\frac{P(q,i)}{P(q)P(i)}\right) . \quad (5)$$

By using an exponential transfer function of the support s_q, we get the posterior probability of the event. Since the independence assumption often is only approximately fulfilled, these equations give only an approximation of the probability. Therefore it is also necessary with a threshold in the transfer function to prevent probability estimations larger than 1,

$$\pi_q = \begin{cases} 1 & s_q \geq 0 , \\ \exp s_q & \text{otherwise} . \end{cases} \quad (6)$$

The measure π_q of the posterior probability of q is also called the *belief value* of the unit.

To use a Bayesian neural network, the units corresponding to observed features are stimulated in such a way that their outputs o_i are set to 1. The activity is then spread through the weights to all other units which sum their inputs according to (3). This sum or support value of each unit is then fed to the transfer function (6), the result of which is returned as an estimation of the posterior probability of the corresponding feature or class. Note that here the activity is propagated only "one step" through the network. The alternative is to let the probability estimations from the first step be input to the next step and iterate until a stable state is reached, i.e. relaxation. This is an important aspect of the Bayesian neural network with many useful applications.[8] However, the best estimation of probabilities is achieved already after the first step and since we are interested in the probabilities in this application, we have here used only one step iteration.

The model also includes negation units (i.e. each attribute has one unit for its existence and one for its nonexistence) in order to make it possible to explicitly represent that an attribute has not been observed. Due to the use of negation units, the belief value of an attribute is calculated as the average between the belief values given by the positive and the negative unit for that attribute (since the independence assumption is only approximately fulfilled, it is not certain that the belief values of the positive and negative units sum to 1):

$$\hat{\pi}_i = \frac{\pi_i + (1 - \pi_{\bar{i}})}{2} . \quad (7)$$

The databases used to train this system consist of a set of examples of the objects to classify where each example consists of the name of the class it belongs to together with a list of (categorical) features. The five databases used below to evaluate the system all contain only one example per class (so-called prototypes) but in Sec. 6 we will treat databases from more realistic situations. The animal database is given in the Appendix since it is the one used in most of the following examples.

Training of the network is done in one pass over the training set during which the probabilities $P(i)$ and $P(i, j)$ are estimated from the training examples. These estimations are then used to set biases and weights according to (4) and (5). Correlated units thus get positive connections between them, anti-correlated units get negative connections and uncorrelated units get zero connections.

The details of the model and its mathematical background are more thoroughly examined in Refs. 8 and 10. Similar neural network models based on probabilities have been studied by e.g. Kononenko[3] and Orponen *et al.*[11]

The Bayesian neural network can in itself be used as a classifier. Some observed features are input to the network and the resulting activity of class units is interpreted as the probability of different classes given the observed features. If the class with the highest probability is chosen, we have an optimal Bayesian classifier.[12] Here, we will augment this network with first an explanatory mechanism and then a question generating system.

3. An Explanatory Mechanism

An important feature of an interactive classification system is the capability to give some explanation of the systems result. If the system is not able to account for its conclusions, it will be harder for people to trust it. In rule based systems, it is possible to achieve simple explanations by showing the user which rules were used for the deduction. In contrast, this is often considered as a problem in neural networks which do not deal with explicit rules.

In the Bayesian neural network model, the units represent external events and the weights between units correlations between these events. Kononenko[13] has shown that it is possible to implement a simple explanatory mechanism for a Bayesian neural network by considering the signals in the network as information gains. Goodman *et al.*[5] presents a method for extracting rules from a Bayesian network also using an information theoretical measure.

Here we will take a similar approach by introducing the *significance* of one event (or unit) for another. We want the significance to be an additive measure such that the total significance for an event from a set of (independent) events is the sum of the significances of the individual events. We also want it to be normalized to 1, in the sense that when a set of independent events make another event completely certain, then their significances for the latter event should sum to 1. A significance of 0 means that an event does not say anything about another. Also, we consider only positive significances. Evidence against a certain event is instead handled as a significance for the negation of the event.

To find such a measure, we note that the support value (3) consists of a sum of the supports from other units. If the support value is transformed to the interval $[0, 1]$, we get

$$\frac{s_q - \beta_q}{-\beta_q} = \sum_i \frac{w_{qi} o_i}{-\beta_q} . \tag{8}$$

From this we can immediately pick out the expression for the significance g_{ji} from unit i to unit j with the above properties:

$$g_{ji} = \frac{w_{ji} o_i}{-\beta_j} . \tag{9}$$

After having stimulated a set of features and received posterior probabilities of the classes, the significance measure can be used to find out which features are most supportive of or contradictory to a certain class, i.e. why a certain class has a high or low probability. Further it is possible to find out the possible future effect of different features, i.e. which features would support a certain class if they were observed, by setting all o_i in (9) to 1 (the *conditional* significance).

The significance measure is used both directly by some of the question generation strategies presented below and in interactive use of the system to allow the user to gain insight into why the current state of the network is reached.

4. Question Generation

When the network is stimulated with information insufficient for a classification, the purpose of the question generation is to find a (not yet stimulated) feature the value of which can add relevant new information for the classification. The neural network together with the question generation can then be used as a query-reply system. Starting with no or very few known features, the question generation decides from the state of the network the best question to ask in the current situation. The reply of the question is then fed into the network (together with all previously known facts) and the question generation is run again on the resulting state of the network. This continues until enough evidence for a reasonably certain classification has been collected.

Since it may be costly to find out the answers, it is preferable that the number of questions required to reach a classification or hypothesis is minimized. Let us therefore first consider the subject of reaching a hypothesis as fast as possible. There is a theoretical lower limit on the average number of questions required (if we want the hypothesis to be correct). This is $\log_2$ of the number of classes which yields exactly if each question can divide the remaining

alternatives in two equally large groups (one that has the feature and one that does not).

Intuitively, in the network this corresponds to asking about a feature with a belief value as close as possible to 0.5. More formally and assuming that each class either has a feature or not, what is to be minimized for each question is the expectancy of the ratio r of the number of possible classes after the question to the number of possible classes before it, i.e. the sum of the ratio of possible classes if the user says "yes" times the probability of the answer "yes" and the ratio of possible classes if the user says "no" times the probability of a negative answer:

$$E(r_i)=\pi_i \cdot \pi_i+(1-\pi_i)\cdot(1-\pi_i)=2\pi_i^2-2\pi_i+1\,, \quad (10)$$

which has a minimum at $\pi_i = 0.5$ and increases from there symmetrically on both sides.

This strategy gave the best results when compared to several other strategies.[14] Figure 1 shows

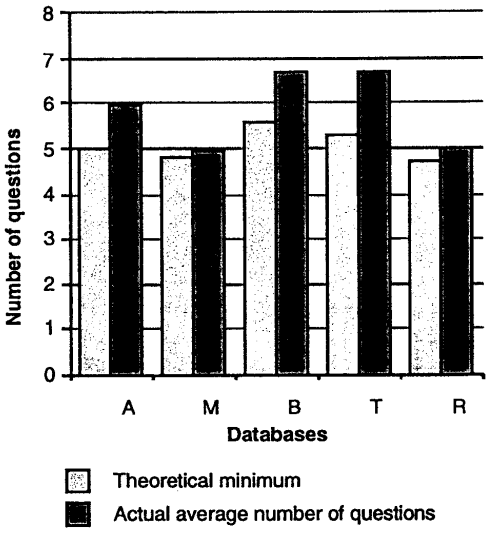

Fig. 1. This diagram shows the average number of questions required to reach a hypothesis for the initial strategy tested on five databases. This is compared to the theoretical minimum, $\log_2$ of the number of classes in each database. The databases are (number of classes in parenthesis): Animals (32), Mushrooms (28), Bumblebees (48), Tree (39) and Random (23). In general, the difference between the theoretical limit and the actual number is less than one question (but for "Tree", the patterns of which are generated from an unbalanced hierarchical tree which makes the $\log_2$-limit impossible to reach).

```
[] Value of land-living? 1
[] Value of eats-grass? 1
(0.1439: rabbit pig kangaroo horse
   giraffe elephant camel antelope)
[] Value of pair-toed? 0
(0.1582: rabbit kangaroo horse elephant)
[] Value of big? 0
(0.2091: rabbit kangaroo elephant)
[] Value of medium? 0
(0.2451: rabbit elephant)
[] Value of small? 1
(0.645: rabbit)
The answer must be rabbit.
```

Fig. 2. This is an example of what a query session might look like with the initial strategy and the animal database. The user is thinking of a rabbit and replies accordingly. "1" means that the animal has the feature, "0" that it does not. The animals which have the highest belief value at a certain stage are shown in parenthesis. Initially nothing in the network is stimulated. About half of the animals in the database are land-living so that is a good initial question. About half of those land-living eats grass and so on. After six questions, "rabbit" has the single highest probability.

how close to the theoretical limit $\log_2$ the number of questions is when this strategy is tried on five different databases. Figure 2 gives an example of what a query-reply session can look like when this approach is used.

In a real application, it might be the case that questions are differently hard or costly to answer. What actually ought to be minimized then is the sum of "costs" of the questions. In such a case, a good strategy is to ask about a feature with as high as possible a value of $\frac{\pi_i-\pi_i^2}{c_i}$ where c_i is the cost to find out about that feature. This is actually an approximation of the formula:

$$-\frac{\log(E(r_i))}{c_i}=\frac{-\log(2\pi_i^2-2\pi_i+1)}{c_i}\,. \quad (11)$$

The justification of this formula comes from the fact that it is equally good to make two questions with costs $= c$ and $E(r_i) = r$ as one question with cost $= 2c$ and $E(r_i) = r^2$. Also note that both this formula and its approximation yields the same strategy as without costs when all the costs are equal.

There is, however, a serious drawback with a strategy that only tries to minimize the number of questions. Due to the lack of redundancy in the questions, it has a very limited fault tolerance.

Usually, each sequence of answers leads to a unique class, so one erroneous input will cause an incorrect classification. Not only is some robustness important to reduce the effect of mistakes in the replies from the user but even more so to make the system robust to ambiguities and uncertainties.

There are plenty of ways to improve the fault tolerance of this system. We have concentrated on two of them, hypothesis verification and inconsistency checking. The idea behind the former is to divide the query session into two phases, ***hypothesis generation*** and ***hypothesis verification***. During ***hypothesis generation***, questions are chosen as described above, i.e. to reach a hypothesis as fast as possible. The network is considered to have a hypothesis when there is only one leading hypothesis and this has a belief value higher than 0.5.

When a hypothesis has been produced, the system will generate *verification* questions. Here, a good strategy is to ask about the most specific feature of the hypothesis, i.e. one possessed by as few as possible of the other classes. This strategy uses the significance g_{ji} defined in the previous section. It asks about the feature that if answered would be most significant for the current hypothesis. If the reply is in accordance with the current hypothesis, this strengthens the network's belief in the hypothesis. If instead the reply does not support the current hypothesis, the network is likely to abandon it and return to the first phase in order to find a better hypothesis.

The dialogue is terminated when a predetermined number (here two) of consecutive verification questions have been answered in accordance with the hypothesis.

Inconsistency checking is also crucial for improving robustness. The purpose is to detect and remove dubious inputs as soon as possible after they have occurred. This is achieved by repeating questions on features for which the user input conflicts with the state of the network and thus may be erroneous. Inconsistency checking is done in both phases although it is more critical in the first where inconsistencies tend to suppress all network activity and therefore may prolong the sequence of questions required to reach a hypothesis.

The search for inconsistencies is done differently in the two phases. Before there is any hypothesis, the strategy looks for contradictory input in general. In this case, an inconsistency is defined as a large difference between the belief value of a feature and what the user has replied on that feature (in these investigations the limit for this difference was 0.3 but this is a completely empirical value).

When the system has a hypothesis, the inconsistency strategy searches for inconsistencies with respect to the specific hypothesis. This is done by searching for strong significances from the user input to the negation of the current hypothesis, i.e. features that contradict the hypothesis.

To sum up, after each new input to the network and calculation of new belief values, the question generation strategies are run as follows. If there is no hypothesis in the network, its state is first searched for a general inconsistency and if none is found, a hypothesis reaching question is generated. If on the other hand, the network has a hypothesis, it is first checked for an inconsistency with respect to the current hypothesis and if none is found a verification question is generated.

It is important to note that all question generation strategies presented here look at the local states of the units in the network and select the unit with maximum value of some function of its state. Some also use the significance presented above which is also a local calculation from the signals in the network. This means that no global calculation has to be done except for the last step which is similar to a "winner-take-all" operation.

5. Empirical Evaluation

We now turn to the performance of these algorithms. First let us look at an example of a query session where the question generation strategies and the explanatory mechanism are used.

We train the network with the animal database and start the system from the beginning without providing any initial information. Say that we see a walrus on the shore and we at first do not realize that it is water-living, i.e. we make a mistake on the very first question:

```
[] Value of land-living?  1
[] Value of eats-grass?  0
[] Value of tail?  0
[] Value of short-tail?  0
[] Inconsistency:
   Value of land-living again?
```

The system has detected a possible inconsistency. Before we answer this question, let us see what

supports or contradicts "land-living" (a minus in front of an attribute denotes the negation of the attribute):

```
Evidence for land-living:
Evidence against land-living:
# 0.57050:  -tail
# 0.26141:  -eats-grass
# 0.22423:  -short-tail
```

The values shown are the significances. (Although in the interval [0, 1] they are not to be confused with probabilities.) All the answers so far speak against "land-living". The strongest evidence against "land-living" is that it has no tail. Suppose that we continue and correct our erroneous answer:

```
[] Inconsistency:
   Value of land-living again?  0
[] Value of biting?  0
[] Value of water-living?  1
[] Value of two-legged?  0
[] Value of brown?  1
[walrus] Value of tusked?
```

Now the system has reached the hypothesis "walrus". Let us see what supports this hypothesis:

```
Evidence for walrus:
# 0.53561:  water-living
# 0.36601:  brown
# 0.21862:  -land-living
# 0.16602:  -tail
Evidence against walrus:
```

To find out why "tusked" is a good verification question, we must know how significant it would be if we answered it. Let us find out the conditional significance for "walrus" from different features:

```
Potential evidence for walrus:
# 0.80000:  tusked
# 0.80000:  very-big
# 0.53561:  water-living
# 0.48301:  eats-animals
# 0.36601:  brown
# 0.21862:  -land-living
# 0.16602:  fur
# 0.16602:  living-offspring
# 0.16602:  -tail
# 0.16602:  -eggs
```

Most significant for "walrus" would be "tusked" and "very-big" which are therefore good verification questions. We continue:

```
[walrus] Value of tusked?  1
[walrus] Value of very-big?  1
The answer must be walrus!
```

Finally we check what supports this conclusion:

```
Evidence for walrus:
# 0.80000:  tusked
# 0.80000:  very-big
# 0.53561:  water-living
# 0.36601:  brown
# 0.21862:  -land-living
# 0.16602:  -tail
Evidence against walrus:
```

Most important for the conclusion "walrus" is in this case "tusked" and "very-big" as expected but also "water-living" and "brown" are quite significant.

In Fig. 3 is shown the average number of questions required as well as the fraction of false

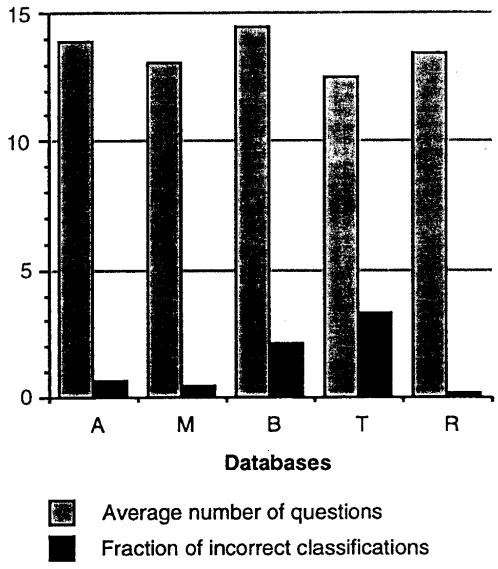

Fig. 3. The average number of questions and the fraction of incorrect classifications (in parts of ten) when all of the question strategies described are used and 10% of the user inputs are disturbed. The databases are the same as in Fig. 1. The fraction of faults is dependent on the structure of the database. "Tree" is a hierarchical tree which is difficult for the system. "Random" is more evenly distributed and the system makes almost no misclassifications. "Animals" and "Mushrooms" are somewhere in between.

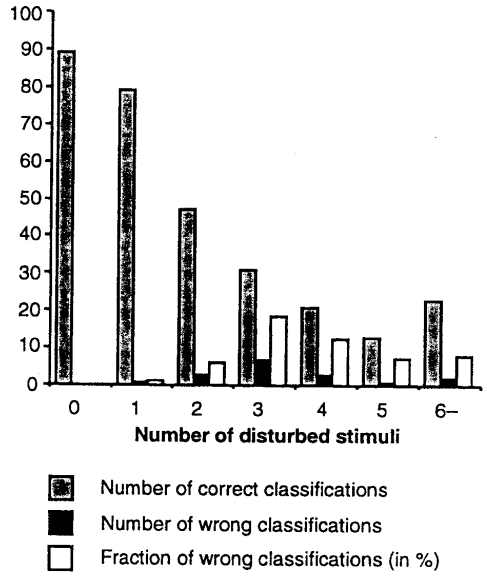

Fig. 4. The grey and black bars show the number of correct and incorrect classifications respectively in our tests, given the corresponding number of disturbances in the reply sequence (the last column contains all cases with six or more disturbances). More interesting is the white bars showing the fraction of incorrect classifications in percent dependent on the number of disturbed inputs. Note that this is not monotonically increasing but seem to have a peak around 3–4 disturbances.

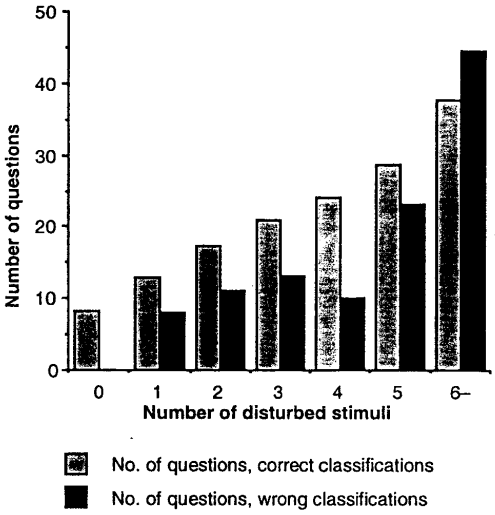

Fig. 5. The average number of questions depending on the number of disturbed inputs, for incorrect and correct classifications separately. When no disturbances are given, the number of questions is about two more than in Fig. 1 corresponding to the two verification questions required. Thereafter every additional disturbance causes about four extra questions, i.e. every question erroneously answered requires three more questions to settle. (Note that the last column contains all cases with six or more disturbances).

classifications when all the strategies above are used. The network was trained with the whole database and then tested with each pattern five times, with 10% of the inputs disturbed.

What is interesting to see is how these two measures, the number of questions and the fraction of faults, depend on the actual number of disturbed inputs. Figure 4 shows for the animal database the number of correct and incorrect classifications together with the fraction of incorrect classifications to the total number. Figure 5 shows the average number of questions for correct and incorrect classifications for the same database. The data for both figures were acquired by testing the animal database ten times, half of them with an average of 10% disturbances and half with an average of 20% disturbances in the inputs.

A more detailed account for the above algorithms and their evaluation can be found in Refs. 15 and 6.

6. More Realistic Databases

The databases used up to this point are all quite simple in their design. Each class has only one example in the training set which means that all objects from the same class have identical features. This is generally not the case in real applications where instead objects from the same class may vary in appearance (i.e. have "fuzzy" features). Furthermore, some features may not be easy to distinguish from each other. As an example, this holds specifically for sizes and colors in the animal database used here. In reality, there are no absolute borders between e.g. "brown", "light brown" and "dark brown" but the neural network does not know about the relations between different colors so if the database states that an antelope is light brown, then it is false to say that it is brown.

One way to handle this, is to associate a real value between 0 and 1 with each feature in each class prototype to reflect how frequent (or how

"certain") that feature is for that class. This will also introduce the relation between features without distinct borders such as colors or sizes in a natural way. An animal that is usually brown might sometimes be light brown and sometimes dark brown (which can be represented in the database by giving "brown" the strength e.g. 0.8, "light brown" 0.2 and "dark brown" 0.2).

In one experiment, the animal database was changed according to this scheme (regarding colors and sizes) and the system tested with it. As can be seen from Fig. 6 the results are nearly identical to those for the original animal database. This indicates that the algorithms proposed are directly applicable to databases of this type as well.

The system has also been tested on some more realistic tasks such as diagnosis of car engines and telephone exchange computers. In the latter case, one malfunctioning circuit card out of 36 is to be identified from an error vector of 122 bits. The database, consisting of 442 examples, is not completely unambiguous (malfunctioning of different cards could give rise to the same symptoms) and in general there are only small differences between the vectors from different fault classes. A system based on a traditional approach manages to classify about half of the examples correctly. The Bayesian neural network manages 75% given all bits in the vector at once. The query-reply system classifies 76% correctly after having asked for, in average, only 36 bits (about 30%) per example. This shows that the system is quite good at picking out just those questions that are relevant for diagnosis.

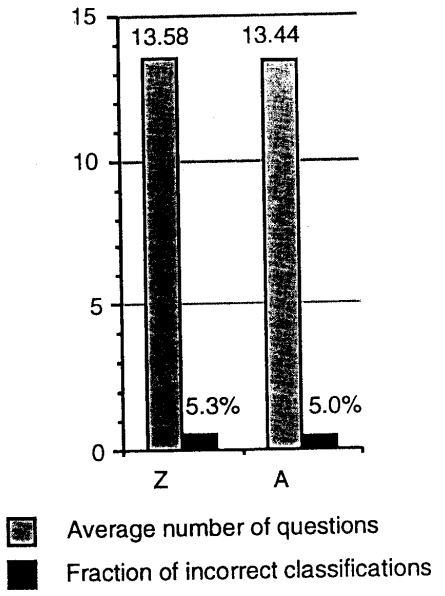

Fig. 6. Results for the "fuzzy" database (denoted by "Z") compared to the results for the normal animal database ("A"). The difference is minimal both for the number of questions and for the fraction of misclassifications.

7. Discussion and Conclusions

This study shows that it is possible to achieve an efficient, robust and flexible query-reply system based on a Bayesian artificial neural network. We claim that a system of the kind presented may be a powerful alternative to rule-based expert systems for classification and diagnosis tasks.

One example of an advantage of this solution over a traditional rule-based one is the natural format of the training sets. Instead of trying to extract some, often artificial, rules that represent the expert knowledge, we just show the neural network a set of examples of the objects to classify together with their labels. Because the artificial neural network does not work with explicit rules, it does not have the same problem as rule-based systems with inconsistencies in the data. Exceptions and ambiguities are handled in a natural way. Further, in spite of not using the rule-based approach, it is possible to get simple explanations from the system in terms of the primary causes for the system's conclusions.

There are a few advantages with the extremely simple interface between the network and the question generation. At each step, all currently known facts are fed into the network after which the question generating strategies decide from the resulting state of the network which is the best question. This means that it is possible to avoid answering a question, to supply some other information not at all related to the question posed, or to change a previous reply. Further, initially providing some known facts makes the query session start from there. In each case, the system will generate the best hypothesis and the best question given the information currently available. This dynamic adaptation of questions to known facts together with the capacity to handle uncertain user input as well as "fuzzy" features in the training set makes the system very flexible.

Here we have only considered attributes in the database that are categorical (even though they may have graded input in terms of probabilities).

However, a natural extension of the model is to allow continuous valued attributes. This can be done via a finite mixture of density functions implementing a kind of soft "interval coding".[10,4,16] Since in many applications there are both continuous valued and categorical attributes, this possibility makes the system interesting for a broader set of tasks.

There are also problems with a one-layer architecture. The one-layer Bayesian neural network model is derived with the assumption of independent attributes. If this assumption is violated, the system may still find the best hypothesis (due to the fault tolerance) but since the estimation of probabilities gets less accurate, the generated questions will be less effective. To overcome this, more layers ("hidden layers") are required in the network. Although not treated here, it is possible to extend the one-layer Bayesian neural network in a natural way to a multilayer architecture.[17,10] This gives the system the capability to efficiently handle situations where the attributes are correlated.

We believe that the approach demonstrated here has a considerable potential for use in many practical applications. Especially the high fault tolerance gives the system the possibility to move out from toy domains to real and noisy environments.

Acknowledgements

This work has been supported by Ellemtel Telecommunication Systems Laboratories (Ellemtel Utvecklings AB) and the Swedish National Board for Industrial and Technical Development (STU-87-01224P). We also want to thank Magnus Stensmo together with whom much of the evaluation of strategies was done.

Appendix: The Animal Database

This is the animal database which is used in the examples (listed in its usual lisp-format).

```
((bat oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   land-living four-legged flying eats-flies very-small grey tail)
(rat oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   gnaw-teeth tail land-living four-legged eats-garbage small brown)
(rabbit oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   land-living four-legged jumping short-tail eats-grass gnaw-teeth very-long-ears small white)
(elephant oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged land-living eats-grass robust big-ears proboscis tusked tail very-big grey)
(horse oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged hoofs eats-grass land-living odd-toed running big brown tail)
(antelope oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged hoofs short-tail eats-grass pair-toed land-living ruminanting lissom antlered running
   medium light-brown)
(giraffe oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged hoofs eats-grass pair-toed land-living ruminanting long-neck tail big yellow)
(camel oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged hoofs eats-grass pair-toed land-living gibbous tail big yellow)
(pig oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged hoofs big eats-grass pair-toed land-living digging tail curl-tail pink)
(walrus oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged eats-animals water-living tusked very-big brown)
(skunk oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged land-living eats-carrion tail smells-terrible medium black)
(hyena oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged land-living long-nosed short-tail eats-carrion medium brown)
(dog oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged eats-animals land-living long-nosed tail lissom medium brown barks)
(bear oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged short-tail eats-animals land-living long-nosed robust big brown)
(lion oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged eats-animals land-living short-nosed tail lissom climbing big yellow)
```

```
(cat oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   four-legged eats-animals land-living short-nosed tail lissom climbing small black)
(ape oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur living-offspring
   land-living short-tail eats-anything two-legged short-nosed human-like big black)
(kangaroo oxygen-consuming moving nervous-system spine CNS blood warm-blooded fur land-living
   four-legged living-offspring pouch jumping eats-grass medium tail light-brown)
(duck oxygen-consuming moving nervous-system spine CNS blood warm-blooded wings nib two-legged
   feathers eggs flying eats-grass small white)
(pelican oxygen-consuming moving nervous-system spine CNS blood warm-blooded wings nib two-legged
   feathers eggs flying eats-fish medium white)
(penguin oxygen-consuming moving nervous-system spine CNS blood warm-blooded wings nib two-legged
   feathers eggs not-flying eats-fish water-living small black)
(ostrich oxygen-consuming moving nervous-system spine CNS blood warm-blooded wings nib two-legged
   feathers eggs not-flying eats-grass running big black)
(crocodile oxygen-consuming moving nervous-system spine CNS blood cold-blooded eggs tail water-living
   four-legged plates eats-animals brown big tail)
(sea-turtle oxygen-consuming moving nervous-system spine CNS blood cold-blooded eggs tail shell
   four-legged eats-grass water-living big brown)
(frog oxygen-consuming moving nervous-system spine CNS blood cold-blooded water-living four-legged
   eggs very-small jumping eats-flies green)
(housefly oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings flying proboscis two-winged fat-body black)
(mosquito oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings flying proboscis two-winged thin-body light-brown)
(butterfly oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings flying proboscis four-winged fat-body yellow)
(beetle oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings flying biting four-winged fat-body black shell)
(dragonfly oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings flying biting four-winged thin-body brown)
(grasshopper oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   trakeas feelers six-legged wings four-winged flying biting jumping fat-body green)
(spider oxygen-consuming moving nervous-system exoskeleton articulations eggs extremely-small
   wingless pipe-trakeas feeler-less eight-legged fat-body black))
```

References

1. R. Davis and D. B. Lenat, *Knowledge-Based Systems in Artificial Intelligence* (McGraw-Hill, New York, 1982).
2. J. R. Quinlan, "Induction of decision trees," *Machine Learning* **1**, 81–106 (1986).
3. I. Kononenko, "Bayesian neural networks," *Biol. Cybern.* **61**, 361–370 (1989).
4. H. G. C. Tråvén, "A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions," *IEEE Trans. Neural Networks* **2:3**, 366–377 (1991).
5. R. M. Goodman, C. M. Higgins, J. W. Miller and P. Smyth, "Rule-based neural networks for classification and probability estimation," *Neural Comput.* **4:6**, 781–804 (1992).
6. M. Stensmo, "A query-reply classification system based on an artificial neural network," Technical Report TRITA-NA-9107, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (1991). Licentiate degree thesis.
7. F. Kozato and Ph. de Wilde, "How neural networks help rule-based problem solving," in *Proc. ICANN-91*, eds. T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (North-Holland, 1991) pp. 465–470.
8. A. Lansner and Ö. Ekeberg, "A one-layer feedback, artificial neural network with a Bayesian learning rule," *Int. J. Neural Syst.* **1:1**, 77–87 (1989).
9. M. L. Minsky and S. A. Papert, *Perceptrons* (MIT Press, 1988).
10. A. Holst and A. Lansner, "A Bayesian neural network with extensions," Technical Report TRITA-NA-9325, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (1993).
11. P. Orponen, P. Floréen, P. Myllymäki and H. Tirri, "A neural implementation of conceptual hierarchies with Bayesian reasoning," Technical Report, Department of Computer Science, University of Helsinki, Finland (1990).
12. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).

13. I. Kononenko, "Bayesian neural network-based expert system shell," *Int. J. Neural Networks* **2:1**, 43–47 (1991).
14. A. Holst, "A comparison between question generation strategies in a query-reply system based on a one-layer neural network," Master's thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (1990). In Swedish, TRITA-NA-E9063.
15. A. Holst, "Further question generation strategies in a query-reply system based on a one-layer neural network," Technical Report TRITA-NA-E9206, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (1992). In Swedish.
16. H. G. C. Tråvén, "On pattern recognition applications of artificial neural networks," PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden (1993).
17. A. Lansner and Ö. Ekeberg, "An associative network solving the "4-Bit ADDER problem," in *Proc. IEEE First Annual Int. Conf. on Neural Networks* (1987) pp. II–549-II-556.