Check for updates

Recognition of Semantically Incorrect Rules: A Neural-network Approach

Li-Min Fu

The University of Wisconsin-Milwaukee
Department of EE & CS
Milwaukee, Wisconsin 53201

ABSTRACT

A novel technique that applies the neural-network learning strategy of back-propagation to recognize semantically incorrect rules is presented. When the rule strengths of most rules are semantically correct, semantically incorrect rules can be recognized if their strengths are weakened or change signs after training with correct samples. In each training cycle, the discrepancies in the belief values of goal hypotheses are propagated backward and the strengths of rules responsible for such discrepancies are modified appropriately. A function called consistent-shift is defined for measuring the shift of a rule strength in the direction consistent with the strength assigned before training and is a critical component of this technique. The viability of this technique has been demonstrated in a practical domain.

1 Introduction

One important issue in designing a knowledge-based system is the management of uncertainty. Among the schemes that have been developed for this purpose, the probability and CF (certainty factor) are most widely used. While a knowledge-based system primarily conducts symbolic reasoning, uncertainty is often handled numerically. A question often raised is how to assign rule strengths (CF's). If we view the rule base as a network and rule strengths as connection weights, it is possible to apply neural-network learning strategies such as back-propagation to assign or adjust rule strengths. It seems that a neural-network learning strategy alone is inadequate for determining rule strengths because it only seeks locally optimum behavior. Rule strengths generated in this way may not carry the desired semantics.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1990 ACM 089791-372-8/90/0007/1013 $1.50

However, it may be a good idea to apply such strategies to refine the knowledge base which for the most part is sound. The complexity analysis of knowledge-base refinement can be found in [7] and [8]. This paper presents the approach of knowledge base refinement using the *back-propagation* strategy.

2 Back-Propagation in Inference Networks

It has been known that a rule-based system (knowledge represented in rules) can be transformed into an inference network where each connection corresponds to a rule, and each node corresponds to a premise or the conclusion of a rule. Reasoning in such systems is a process of propagating and combining multiple pieces of evidence through the inference network until final conclusions are reached. Uncertainty is often handled by adopting the certainty factor (CF) or the probabilistic schemes which associate each fact with a number called *belief value*. An important part of reasoning tasks is to determine the belief values of predefined final hypotheses given the belief values of observed evidence. The network of an inference system through which belief values of evidences or hypotheses are propagated and combined is called *belief network*. Correspondence in structural and behavioral aspects exists between neural networks and belief networks. For instance, the summation function in neural networks corresponds to the function for combining certainty factors in MYCIN-like systems. The thresholding function in neural networks corresponds to predicates such as SAME (in MYCIN-like systems) which cuts off any certainty value below .2. Fu [5] described an approach that maps a rule-based system such as MYCIN [1] into a neural architecture.

When a rule-based system makes error, a key issue is how to identify and correct rules responsible for these errors. An error can be defined as the disagreement be-

tween the belief value generated by the system and that indicated by a knowledge source assumed to be correct (e.g., an expert) with respect to some fact. The problem of identifying the sources of errors is known as the *blame assignment* problem. TEIRESIAS [2] is the typical work. It maintains the integrity of the knowledge base by interacting with experts. However, as the size of the knowledge base grows, it becomes difficult for human experts to consider all possible interactions among knowledge in a coherent and consistent way. TMS [3] resolves inconsistency by altering a minimal set of beliefs, but it lacks the notion of uncertainty in the method itself.

Back-propagation [6] is a powerful technique to train the neural network. It is a recursive heuristic which propagates backwards errors from a node to all nodes pointing to that node, and modifies the weights of connections leading into nodes with errors. This technique is just a kind of gradient-descent technique which minimizes a given criterion function iteratively. If we define the criterion to be the difference (or the square of the difference) between the desired and the actual belief values of goal hypotheses, we can modify the strengths of rules inferring these hypotheses by the back-propagation procedure. However, there are important differences between the knowledge-based network and the neural network. The former is often much more sparse than the latter since many connections have no psychological meanings. In addition, rule strengths in the knowledge-based network before training can be estimated from a knowledge source, whereas in the neural network, weights are randomized before training. These differences explain why the knowledge-based network can be kept from some annoying problems that are encountered by the neural network. The sparseness of the knowledge-based network and knowledge-based decomposition of a network into a number of independent networks can help escape the problem of combinatorics. The knowledge-based assignment of rule strengths before training can facilitate convergence to a desired result.

On a given trial, the network generates an output vector given the input vector of the training instance. The discrepancy obtained by subtracting the network's from the desired output vector serves as the basis for adjusting the strengths of connections involved. The adapted *back-propagation* rule is formulated as follows:

$$\triangle W_{ji} = rD_j(\partial O_j/\partial W_{ji}) \quad (1)$$

where

$$D_j = T_j - O_j,$$

$\triangle W_{ji}$ is the weight adjustment of the connection from input node i to output node j, r is a trial-independent learning rate, D_j is the discrepancy between the desired belief value (T_j) and the network's belief value (O_j) at node j, and the term dO_j/dW_{ji} is the derivative of O_j with respect to W_{ji}. According to this rule, the magnitude of weight adjustment is proportional to the product of the discrepancy and the derivative above.

A multi-layered network involves at least three levels: one level of input units, one level of output units, and one or more levels of hidden units. Learning in a multi-layered network is more difficult because the behavior of hidden units is not directly observable. Modifying the weights of connections pointing to a hidden unit requires the knowledge of the discrepancy between the network's and the desired belief value at the hidden unit. However, the desired values at hidden units are not given. The discrepancy at a hidden unit can be derived from the discrepancies at output units which receive activation from the hidden unit [6] This is a recursive definition in which the error at a hidden unit is always derived from errors at the next higher level. Hence, errors are propagated backwards.

In addition, the network's belief value at a hidden unit can be obtained by propagating the belief values at input units recursively and combining these values properly until the hidden unit is reached.

The mathematical requirement for applying backpropagation is that the relationship between the output and the input of a node is determined by a differentiable function. One apparent difficulty with the use of the back-propagation procedure in symbolic inference networks is that combining belief values in most cases involves such logic functions as logic AND and logic OR. To handle belief values, logic AND returns the minimum of its arguments, while logic OR returns the maximum. Since these logic functions are not differentiable in the whole domain, backpropagation cannot be applied directly. This problem can be solved if we view backpropagation as performing hill-climbing search rather than gradient descent search. Because of space limitation, this approach will not be described here. Interested readers can refer to [5].

3 Recognition of Semantically Incorrect Rules

Here, *semantically incorrect* rules refer to those rules that are inconsistent with the semantics defined by the desired local or global inference behavior of the system. Such rules result from inadvertence or misconception of knowledge engineers or experts and may cause the system to reach incorrect conclusions. System performance can usually be improved by removing these rules.

Semantically incorrect rules are classified into three types. In the first type, a rule has a positive CF but its premise actually disconfirms its action. In the second type, a rule has a negative CF but its premise actually confirms its action. In the third type, a rule has a positive or negative CF but its premise neither confirms nor disconfirms its action.

Although semantically incorrect rules can often be discovered by scrutinizing the rule base or by running on test cases, it is not always an easy task especially when the rule base is large. A technique that allows one to recognize this kind of rules will be useful in the course of knowledge engineering.

Suppose a system has reached an equilibrium. If the system loses the equilibrium because of some perturbation and then reaches another equilibrium again, the new equilibrium will usually be close to the old one unless the perturbation is great. Furthermore, if we assume that the system equilibrium depends on a number of variables, the variables under greater perturbation will be subject to greater changes than those under smaller perturbation so that the new equilibrium can be attained faster and be closer to the old one. This intuition also applies to the following argument. Suppose we have a neural network that has been trained to converge to a stable state. If we deliberately change the weights of a small number of connections and then train the network again with the same set of samples, we can expect that the network will be restored to the state close to the previous state and those perturbed connections will have greater changes in weights than unperturbed ones.

This argument can be further extended. Suppose we add some connections to a neural network that has already arrived at an equilibrium and assign weights to these added connections in such a way that incorrect output vectors are generated. Thus, these connections as a whole are semantically incorrect. Then, if we train the network with correct samples, the weights of the added connections will be modified in the direction of minimizing their effect. What happens is that the weights will go toward zero and even cross zero during training.

Recall that three types of semantically incorrect rules have been defined. Consider the connection pointing from node A to node B. If the activation at node A is positively correlated with that at node B in the statistical sense, then a negative weight is semantically incorrect. Likewise, if the activation at node A is negatively correlated with that at node B, then a positive weight is semantically incorrect. Thirdly, when the activation at node A is uncorrelated with that at node B, a significant nonzero weight either positive or negative is semantically incorrect. Suppose we train the neural network with correct samples. In the first two cases, the weight is expected to first shift toward zero and then cross zero; in the third case, the weight is expected to gradually approach zero.

The notion of *consistent shift* for connection weights is introduced as follows. If the absolute magnitude of a weight after training is greater than or equal to that of the weight before training and their signs are the same, then the weight shift is said to be semantically consistent with the weight before training; otherwise the shift is inconsistent. The function *consistent-shift* is defined by

$$\text{consistent-shift} = \begin{cases} w_a - w_b & \text{if } w_b > 0 \\ w_b - w_a & \text{if } w_b < 0 \\ |w_a - w_b| & \text{if } w_b = 0 \end{cases}$$

where w_a and w_b denote the weights after and before training respectively. With this definition, a shift of weight is consistent if *consistent-shift* is greater than or equal to zero; else it is inconsistent.

While semantically incorrect rules will generally not experience consistent weight shift by training with correct samples, the weight shift of semantically correct rules is not always consistent. If the weight assigned to a semantically correct rule before training is overly high, inconsistent weight shift may be observed after training. However, if the weight before training is sufficiently accurate, the degree of inconsistent shift cannot be great. Moreover, when semantically correct rules are mixed with some incorrect rules, the weights of correct rules will often be reinforced consistently to mitigate the effect of incorrect rules. Therefore, under such circumstance, the chance and the degree of inconsistent shift for semantically correct rules will be even smaller. In practice, a weight shift is regarded as inconsistent only if the corresponding value of the function *consistent-shift* is less than a predetermined negative threshold. In the experiments conducted, -.2 was used as the threshold. Besides, because of the reinforcement in the weights of correct rules, the weights of incorrect rules do not necessarily move across or approach zero by training.

When inconsistent weight shift is observed, the corresponding weight before training is considered as semantically incorrect. However, an important assumption is that the network is trained with an initial state where the weights of most connections are semantically correct rather than randomly chosen and are not overly high. If initial weights are randomly chosen, it makes no sense to state whether a shift is consistent with the initial assignment or not. Furthermore, since the back-propagation rule is just a gradient-descent algorithm seeking a local optimum, random weights before training will most likely lead to post-training weights that are dissociated from the semantics normally embedded in corresponding rules. Hence, it is essential that most of rules are

semantically correct so that learning can be constrained properly.

The procedure for recognizing semantically incorrect rules is given below:

- *step 1.* Train the network derived from a rule base using the back-propagation procedure.
- *step 2.* Compute the value of the function *consistent-shift* for each rule after training.
- *step 3.* If the value of *consistent-shift* is less than a predetermined negative threshold, then put the rule associated with its CF before training in the set of semantically incorrect rules.

The output can further be examined by human experts. The procedure applies to a collection of rules and is not affected by the order of rule acquisition.

When training samples are not described by intermediate concepts (which is often the case), a procedure based on sample statistics is not feasible for debugging intermediate rules. In the above procedure, the back-propagation heuristic allows the derivation of local errors manifested at hidden nodes from global errors at output nodes and is therefore important for detecting semantically incorrect rules involving intermediate concepts.

Example 1

A rule base is given which involves five binary-valued features, three intermediate hypotheses and three final hypotheses. Assume that the three final hypotheses are mutually exclusive but the three intermediate hypotheses are not. The training samples are displayed in Table 1. The rule base consisting of Rule-B001 .. Rule-B0015 shown in Figure 1 can classify or diagnose these samples correctly. To make sure that the strength of each rule in this rule base is proposely assigned, the rule base was trained with the samples; the resulting weights are shown under the category "Before training" in Table 2. In the next step, six rules were added including Rule-B0016 .. Rule-B0021 (see Figure 1) to the rule base. The weights assigned to the six rules are also shown under the category "Before training" in Table 2. These added rules are semantically inconsistent with the training samples since only 10 out of 20 samples are classified correctly using the new rule base.

The network derived from the rule base composed of Rule-B001 .. Rule-B0021 was then trained with the samples. The resulting weights upon convergence (at cycle 3000) were recorded and are shown under the category "After training" in Table 2. The value of the function *consistent-shift* was computed for each rule and is also included in Table 2. Assume that the threshold for determining semantically incorrect rules is -.2. All the six

Table 1: Training samples

E_1	E_2	E_3	E_4	E_5	C_1	C_2	C_3
no	no	yes	no	no	yes	no	no
no	no	yes	no	yes	yes	no	no
no	yes	no	no	yes	yes	no	no
no	yes	no	yes	yes	no	no	yes
no	yes	yes	no	no	no	yes	no
no	yes	yes	no	yes	yes	no	no
no	yes	yes	yes	no	no	yes	no
yes	no	no	no	no	no	no	yes
yes	no	no	no	yes	no	no	yes
yes	no	no	yes	no	no	no	yes
yes	no	no	yes	yes	no	no	yes
yes	no	yes	no	no	yes	no	no
yes	no	yes	no	yes	yes	no	no
yes	no	yes	yes	yes	no	no	yes
yes	yes	no	no	yes	yes	no	no
yes	yes	no	yes	yes	no	no	yes
yes	yes	yes	no	no	no	yes	no
yes	yes	yes	no	yes	yes	no	no
yes	yes	yes	yes	no	no	yes	no
yes	yes	yes	yes	yes	yes	no	no

Rule-B001: $M_1 \longrightarrow C_1$
Rule-B002: $M_3 \longrightarrow C_1$
Rule-B003: $M_1 \longrightarrow C_2$
Rule-B004: $M_2 \longrightarrow C_2$
Rule-B005: $M_2 \longrightarrow C_3$
Rule-B006: $M_3 \longrightarrow C_3$
Rule-B007: $E_2 \longrightarrow M_1$
Rule-B008: $E_3 \longrightarrow M_1$
Rule-B009: $E_4 \longrightarrow M_1$
Rule-B0010: $E_1 \longrightarrow M_2$
Rule-B0011: $E_2 \longrightarrow M_2$
Rule-B0012: $E_3 \longrightarrow M_2$
Rule-B0013: $E_2 \longrightarrow M_3$
Rule-B0014: $E_4 \longrightarrow M_3$
Rule-B0015: $E_5 \longrightarrow M_3$

Rule-B0016: $M_2 \longrightarrow C_1$
Rule-B0017: $M_3 \longrightarrow C_2$
Rule-B0018: $M_1 \longrightarrow C_3$
Rule-B0019: $E_1 \longrightarrow M_1$
Rule-B0020: $E_5 \longrightarrow M_2$
Rule-B0021: $E_3 \longrightarrow M_3$

Figure 1: Inference rules

Table 2: Rule strengths before and after training

Rule name	Before	After	Consistent-shift
Rule-B001	.9	.95	0.05
Rule-B002	-.9	-.9	0
Rule-B003	.9	.7	-.2
Rule-B004	-.8	-.85	0.05
Rule-B005	.7	.8	.1
Rule-B006	-.7	-.85	.15
Rule-B007	.8	.8	0
Rule-B008	.7	.8	.1
Rule-B009	-.4	-.7	.3
Rule-B0010	.6	.65	0.05
Rule-B0011	.3	.3	0
Rule-B0012	-.5	-.35	-.15
Rule-B0013	.8	.75	-0.05
Rule-B0014	-.3	-.3	0
Rule-B0015	-.6	-.67	0.07
Rule-B0016	.7	.18	-.52
Rule-B0017	-.6	-.65	0.05
Rule-B0018	.5	-.6	-1.1
Rule-B0019	.4	.15	-.25
Rule-B0020	-.6	-.38	-.22
Rule-B0021	.5	.18	-.32

added rules except Rule-B0017 can be recognized as semantically incorrect. After excluding the five recognized rules, the misclassification rate becomes 3/20 with an improvement of 10/20 - 3/20 = 7/20. The failure to recognize Rule-B0017 as incorrect can be ascribed to the uncertainty involved in this example. Rule-B003 with consistent-shift = -.2 is very close to being recognized as an incorrect rule. From Figure 1 and Table 2, we see that Rule-B001 and Rule-B003 are inconsistent in the sense that C_1 and C_2 are mutually exclusive. Thus, it is reasonable to believe that either Rule-B001 or Rule-B003 is semantically incorrect. From the consistent-shift, Rule-B003 should be the incorrect one. If Rule-B003 is removed from the original rule base including Rule-B001 .. Rule-B0015, the remaining rules can still classify all the samples correctly.

4 Results on a Practical Domain

The back-propagation procedure has been experimented with in the domain of diagnosing jaundice. To handle the problem due to logic conjunction, the first approach, called approach I, uses approximate differentiable functions, whereas the second approach, called approach II, uses hill-climbing search (see [5]). The algorithms were implemented in COMMONLISP.

Table 3: The number of incorrect rules with consistent-shift greater than -.2 (called index A) and the number of correct rules with consistent-shift less than -.2 (called index B) after applying the back-propagation technique under approach I and II.

Incorrect rule no.	Approach I		Approach II	
	Index A	Index B	Index A	Index B
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	2	0	0
6	0	2	0	0
7	1	2	0	0
8	1	3	1	1
9	2	4	1	1
10	3	4	2	2

A rule base, derived from JAUNDICE [4], contained 50 rules, 5 diseases, 5 intermediate hypotheses, and 15 clinical attributes. This rule base was transformed into a four-layered network with 5 output units, 28 hidden units, and 15 input units.

Ten experiments were carried out. In each experiment, a small number of incorrect rules (the definition was given earlier) were added in the network described above. In order to evaluate how good errors can be propagated to hidden layers, most of the incorrect rules added were connected with hidden units in each experiment.

Twenty instances were used as training samples. All these instances can be diagnosed correctly with the 50 rules and were collected from the JAUNDICE case library where cases were obtained from the Stanford Medical Center. The frequencies of the five diseases were equal.

As shown in Table 3, approach I had more false detections than approach II. A correct rule sometimes experienced a significant inconsistent shift of the weight after the procedure because of its interaction with incorrect rules. Fortunately, when this was the case, the correct rule was often decided to be kept because its removal would worsen the system performance. If an incorrect rule contributes to the error observed, it will generally be identified; otherwise, it may not be detected. This is the limitation of the back-propagation learning technique.

5 Conclusion

A physical system at an equilibrium will tend to maintain that equilibrium when undergoing small perturbation. Likewise, when a neural network is moved away from an established optimum state, it will tend to restore (relax toward) that state. This observation is the rationale behind the approach presented in this paper.

By semi-qualitatively reasoning with the shift of rule weights after training with correct samples, semantically incorrect rules can be recognized. This technique is particularly useful for debugging intermediate rules when we only have samples that are not described by any intermediate concepts. However, it is important to point out that the sparseness of the knowledge-based network and knowledge-based decomposition of a network into a number of independent networks can alleviate the problem of combinatorics (difficulty in scaling up) that often arises in the neural-network approach. In addition, there must be adequate initial knowledge to make the process less random and more predictable.

References

1. Buchanan, B.G. and Shortliffe, E.H., Rule-Based Expert Systems, Addison-Wesley, Massachusetts, 1984.

2. Davis, R., Application of meta-level knowledge to the construction, maintenance, and use of large knowledge base, Ph.D. thesis, Computer Science Dept., Stanford U., 1976.

3. Doyle, J., A truth maintenance system, *Artificial Intelligence*, 12(3), 1979, 231-272.

4. Fu, L.M., Learning object-level and meta-level knowledge in expert systems, Ph.D. thesis, Stanford U., 1985.

5. Fu, Li-Min, Integration of neural heuristics into knowledge-based inference, *Connection Science*, 1(3), 1989, 327-342.

6. Rumelhart, D.E., Hinton, G.E. and Williams, R.J., Learning internal representation by error propagation, In Parallel Distributed Processing: Explorations in the Microstructures of Cognition, MIT press, Cambridge, 1986.

7. Valtorta, M., Some results on the complexity of knowledge-base refinement, in Proceedings of IWML-6, Morgan Kaufmann, 1989, 326-331.

8. Wilkins, D.C. and Buchanan, B.G., On debugging rule sets when reasoning under uncertainty, in Proceedings of AAAI-86, Philadelphia, 1986, 448-454.