Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006

NON-MONOTONIC REASONING WITH CONNECTIONIST NETWORKS USING COARSE-CODED REPRESENTATIONS

SRIRAM .G. SANJEEVI[1], PUSHPAK BHATTACHARYYA[2]

[1]Department of Computer Science & Engg., National Institute of Technology, Warangal 506004, India
[2]Department of Computer Science & Engg., Indian Institute of Technology, Bombay, India
E-MAIL: sgsanjeevi@yahoo.com, pb@cse.iitb.ac.in

Abstract:

This paper, describes a connectionist fault-tolerant non-monotonic reasoning system, which uses coarse-coded distributed representations. Distributed representations are known to give the advantages of fault tolerance, generalization and graceful degradation of performance under noise conditions. A semantic network is designed, using a novel approach, with connectionist networks using coarse-coded representations to perform non-monotonic reasoning. The system performs non-monotonic reasoning using the property of inheritance. The system also supports the feature of cancellation of inheritance, whereby more specific information associated with the nodes lower in the 'isa' hierarchy is given precedence over default information associated with the nodes higher in the hierarchy. System has exhibited good generalization ability on unseen test inputs. System's performance with regard to its ability to exhibit fault tolerance under noise conditions is also studied. The system offers very good results of fault tolerance under noise conditions.

Keywords:

Connectionist; coarse-coded; fault-tolerance; non-monotonic; reasoning; semantic-network

1. Introduction

Traditionally reasoning systems using predicate logic have been implemented using symbolic methods of artificial intelligence. Connectionist methods of implementation of reasoning systems describe an alternative paradigm. There are two types of reasoning 1) Monotonic and 2) Non-monotonic. In a monotonic reasoning system adding new axioms to the system does not diminish the set of theorems that can be proved. In non-monotonic reasoning systems we make default inferences. These default inferences are assumed to be true in the absence of more specific information. If more specific information becomes available, the default inferences, which are contradictory with the specific information, will be with drawn.

Among the connectionist systems two types of representational schemes can be used. These are 1) localist and 2) distributed representational schemes. Localist representational schemes represent each concept with an individual unit or neuron. In the distributed representational schemes, each unit or neuron is used in representation of multiple concepts and multiple units or neurons are used to represent a single concept. In the literature, some localist methods for reasoning using connectionist networks have been described. The connectionist inference system *SHRUTI* [1, 2], 3] described a localist method where temporal synchrony was used to create bindings between variables and entities they represent. A variable *x* of the predicate *give (x, y, z)* is getting bound to an entity *d* if the nodes representing them fire during the same phase of time *p1* during the predicate *p* activation period *T*. The time period *T* is divided into three phases *p1, p2* and *p3* during which synchronous firing of variables *x, y* and *z* and entity nodes they bound respectively takes place. This method has used temporal synchrony as a mechanism to establish variable binding. *CONSYDERR* [4] described a localist method for variable binding and forward reasoning. It uses an assembly or a set of interconnected nodes to represent each predicate *p* (x_1... *xk)*. Each assembly contains one C node for storing the confidence value of the predicate *p* and *k X* nodes to store the binding values for *k* variables of the predicate *p*. A separate node is allocated for each variable of a predicate. Each such node, stores a value representing a particular object being bound with that variable. Different objects that can get bound to a variable will be given separate values. Both these systems used localist representations for the instantiated predicates and performed predicate logic reasoning. In our earlier works [5, 6], and [7] we proposed and described predicate logic reasoning systems using neural networks which used coarse-coded distributed representations to represent instantiated predicates. In these works, we have described

the advantages of using the coarse-coded representations along with neural networks for doing predicate logic reasoning. The predicate logic reasoning is categorized as *monotonic reasoning*. Here, we propose to examine the use of the coarse-coded distributed representations along with the neural networks in the implementation of a non-monotonic reasoning system. A non-monotonic reasoning system makes default inferences in the absence of more specific information. It is investigated here, in this work, to examine the advantages gained by the non-monotonic reasoning system by using the distributed coarse-coded representations in a connectionist framework. Our motivation, is to make available, the advantages of distributed coarse-coded representations to the non-monotonic reasoning system. We also deal with the issue of how to override the default inferences, if more specific information became available in the system.

2. Semantic network

We implement the semantic network [8] shown in figure 1 to perform non-monotonic reasoning. A semantic network consists of a number of labeled nodes interconnected with labeled arcs. In the figure 1 each of the intersecting points between two or more lines is a *node*. The nodes are shown as dots in the fig.1.The line connecting any two nodes in the network is an *arc*. These arcs represent relationship between those nodes. The relationship is shown as a label on the arc. In addition, from each of the nodes there are one or more labeled arcs which do not terminate in any node. The label on each of these arcs indicates the attribute name associated with that node. Each of these arcs terminates with an attribute value associated with the attribute mentioned on the labeled arc. For example, the node *canary* has two labeled arcs *can* and *is*. These arcs are terminated with attribute values *sing* and *yellow* respectively. Hence, the node canary is associated with the attribute values *sing* and *yellow* corresponding to the attributes *can* and *is* respectively. In addition, node *canary* inherits the attributes and corresponding attribute values of nodes higher in the hierarchy of the semantic network through the *isa* and *instance* links. Thus, it inherits attributes and corresponding attribute values of nodes *bird, animal* and *living_thing*. Hence, by the mechanism of inheritance, it is inferred that *canary* is a *bird* and that it can *fly*, *canary* is an *animal* and that it can *move* and *canary* is a *living thing* and that it can *grow*. Making inferences, by using property inheritance as described here is called *non-monotonic reasoning*.

Our task is to construct the semantic network by using connectionist networks which use coarse-coded representations for information associated with each of the nodes in the semantic network and perform non-monotonic reasoning successfully by using the system.

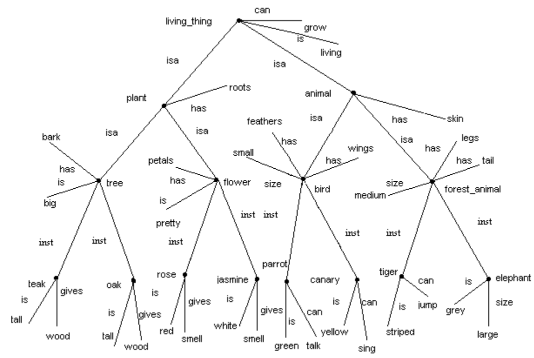

Figure 1. Semantic Network

2.1. Description of data used by neural networks in the system

Table 1. Shows a sample of input data for Neural Network 1

bird 'id' code	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
Localist code representing attribute 'can'	0 0 0 0 1
Localist code representing attribute 'is'	0 0 0 0 1
Localist code representing 'inst'	0 0 0 0 1

The nodes *canary, parrot* are members of the class *bird*. These relationships are indicated by *instance* links in figure 1. The nodes *bird* and *forest_animal* are sub-classes of the class *animal*. These relationships are indicated by *isa* links. Class *animal* is a sub-class of class *living_thing*. This relationship is also indicated by an *isa* link in the figure 1. In *Tables 1* and *2* below we show samples of localist vectors that can be given as inputs and outputs of neural

network 1. Localist representations represent each concept with an individual unit or neuron.

Since, this is a localist representation, the *bird id* code for different birds, will have a *1* at a different position in the code to identify the birds, in table 1. Attributes *can, is* and the *inst* link are each represented by a *1* in the 5^{th} position of the 5 bit codes.

Table 2. Shows a sample of output data for Neural Network 1

Localist Value for attribute 'can'	0 0 0 0 0 0 0 0 0 1
Localist Value for attribute 'is'	0 0 0 0 0 0 0 0 1
Localist Value for 'inst'	0 0 0 0 1

In table 2, the localist representation for the *value* of attribute *can* is *0000000001* and this encodes *talk*, localist representation for the *value* of attribute *is* is *000000001* and this encodes *green* and the localist representation for the *value* of *inst* is *00001* and it encodes *bird*. When given the input vector describing the identification code for a specific bird *parrot* and codes for attributes *can*, *is* and for the membership link *inst, neural network 1* can be trained to produce on its outputs the codes representing the *values* for the above attributes *can*, *is* and the *inst* link namely *talk*, *green* and *bird* respectively. This is a representation for information that *parrot* can *talk*, is *green* and is an instance of *bird*. Since, it is our objective to design and implement a semantic network with connectionist networks using coarse-coded representations, we convert the localist vectors shown above in *tables 1* and *2* into coarse-coded vectors and use them to train the *neural network 1.*

Below, we explain the process of obtaining the coarse-coded vectors from the localist vectors.

2.2. Obtaining coarse-coded distributed representations

Consider the following input localist vector from the table 1.

'0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1'

We view the above vector as being kept in overlapping coarse zones of length of *4* consecutive bits and encode the zone as *1* if there is at least one *1* bit in that zone or else as *0*. We then consider next coarse zone and encode it as *1* or *0* following the above method. We do this process left to right starting from the left most bit. We do this encoding process for the above localist vector to get the following coarse-coded vector.

'0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1'

Coarse-coding can be applied when the number of *1*'s in the original string is sufficiently sparse. If the number of *1*'s in the original string is not sufficiently sparse, then the coarse-coded string when decoded will not yield the original string. This is the reason, we have chosen a 5 bit string to denote each of the attributes *can, is* and *inst* (in which first 4 bits were kept as zeros) for the localist representation of input data of *neural network 1*. Coarse-coding increases the information capacity [9] by increasing the number of units active at a time compared to localist codes which have sparsely populated *1*'s. The amount of information conveyed by a unit that has a probability '*p'* of being '*1'* is

$$-p \log(p) - (1-p) \log(1-p).$$

We obtain the coarse-coded representations of input and output vectors for all the neural networks used in the system, using the above described method. We show here a sample of the coarse-coded representations of the input and output vectors for *neural network 1*.

Table 3. Shows a sample of coarse-coded input data for Neural Network 1

bird 'id' code	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
Coarse-coded representation for attribute 'can'	0 1 1 1 1
Coarse-coded representation for attribute 'is'	0 1 1 1 1
Coarse-coded representation for attribute 'inst'	0 1 1 1 1

Table 4. Shows a sample of coarse-coded output data for Neural Network 1

Coarse-coded value for attribute 'can'	0 0 0 0 0 0 1 1 1 1
Coarse-coded value for attribute 'is'	0 0 0 0 0 1 1 1 1
Coarse-coded value for attribute 'inst'	0 1 1 1 1

3. Organization of Neural Networks in the Connectionist Non-monotonic Reasoning System

The neural networks 1 to 8 in figure 2 accomplish the non-monotonic reasoning using the coarse-coded vectors. They generate non-monotonic inferences by using the mechanism of forward reasoning. Consider the *neural network1* shown in figure 2. When impressed on its inputs, a coarse-coded vector $\mathbf{v_b}$ containing the identification code for a bird, say *parrot* and codes for attributes *can*, *is* and membership link *inst, neural network 1* produces on its outputs the coarse-coded representations for *talk, green* and *bird* respectively. The codes for *talk* and *green* are shown as *v1* and *v2* on the outputs of *neural network 1*. This is a representation for information that *parrot* can *talk*, is *green* and is an instance of *bird*. The coarse-coded representation for *bird* gets impressed on *neural network 6* inputs along with codes for attributes *has*, *has* and *size.* This generates on the outputs of *neural network 6* coarse-coded representations for *feathers, wings, small* and *animal* respectively. The codes for *feathers, wings* and *small* are shown as *v1, v2* and *v3* on the outputs of *neural network 6*. The coarse-coded representation for *animal* gets impressed on *neural network 7* inputs along with the code for attribute *has.* This generates on the outputs of *neural network 7* coarse-coded representations for *skin* and *living_thing* respectively. The code for *skin is* shown as *v1* on the outputs of *neural network 7*. The coarse-coded representation for *living_thing* gets impressed on inputs of *neural network 8* along with the code for attributes *can* and *is*. This generates on the outputs of *neural network 8* coarse-coded representations for *grow* and *living* respectively. The codes for *grow* and *living* are shown as *v1* and *v2* on the outputs of *neural network 8.*

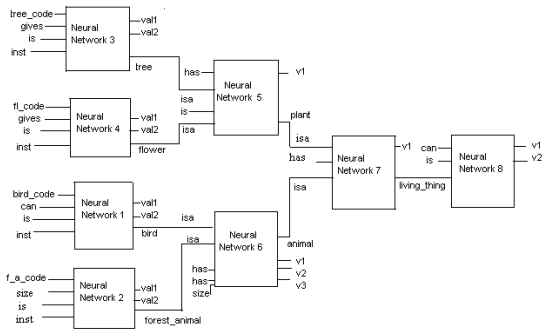

Figure 2. Neural networks for performing non-monotonic reasoning

By noting, the inputs and outputs of neural networks 1, 6, 7 and 8 we obtain the information that *parrot* is a *bird*, it can *talk*, it is *green*, it has *feathers* and *wings*, it is an *anima*l, it has *skin* and also it is a *living thing* and it can *grow* and it is *living*. Both the information explicitly associated with the node *parrot* and the information that can be obtained through property inheritance in the semantic network was successfully obtained by the connectionist reasoning system.

3.1. Cancellation of inheritance

We have performed *non-monotonic reasoning* while implementing a semantic network in a connectionist environment using *coarse-coded representations.* A non-monotonic reasoning system makes default inferences in the absence of more specific information. In the semantic network shown in the figure 1 node *forest_animal* is associated with the attribute *size.* The value of this attribute is indicated as *medium*. However, the node *elephant* which is lower in the *isa* hierarchy has the attribute *size* associated with the value *large*. Here, we need to override the default inference that the *size* of the *elephant* is *medium,* with the more specific information available lower in the *isa* hierarchy, that the *size* of the *elephant* is *large*. This reasoning mechanism is referred to as the *cancellation of inheritance*. In our connectionist reasoning system, we override the information associated with an entity from *higher* in the *isa* hierarchy, with the relevant information available *lower* in the *isa* hierarchy. Therefore, we associate *elephant* with the size *large*. This information is available to the node *elephant* at the lowest point in the *isa* hierarchy, since the node *elephant* is directly connected to this value through the attribute arc *size*.

4. Testing

Following, are the details of the neural networks used to do the above mentioned work. The neural networks in table 5 are feed forward neural networks [10] using the back-propagation algorithm.

Table 5. Shows the details of neural networks used

Neural Network	No. of input units	No. of hidden units	No. of output units
1	126	70	24
2	126	70	24
3	100	60	24
4	100	60	24
5	20	18	15
6	25	27	30
7	10	8	10
8	15	12	10

The non-monotonic reasoning tasks were successfully accomplished to give the expected results.

Table 6. Shows the details of test 1

	No. of Training Patterns	No. of Test Patterns	No. of Patterns Corrected
Neural Network 1	108	108	105

Table 7. Shows the details of test 2

	No. of Training Patterns	No. of Test Patterns	No. of Patterns Corrected
Neural Network 4	81	81	80

Table 8. Shows the details of test 3

	No. of Training Patterns	No. of unseen test patterns	No. of patterns correctly generalized
Neural Network 1	72	36	32

Table 9. Shows the details of test 4

	No. of Training Patterns	No. of unseen test patterns	No. of patterns correctly generalized
Neural Network 4	51	30	30

Secondly, the performance of the above coarse-coded non-monotonic reasoning system was tested for error tolerance under noise conditions. In the test *1*, *neural network 1* with *126* input units, *70* hidden units and *24* output units was trained with *108* patterns. These, were made test patterns after introducing *1* bit error at a random location in each of these patterns. In the test *2*, *neural network 4* with *100* input units, *60* hidden units and *24* output units was trained with *81* patterns. These *81* patterns were made test patterns after introducing *1* bit error at a random location, in each of these patterns. Results are as shown in tables *6* and *7*. The coarse-coded non-monotonic reasoning system was found to be highly fault tolerant to errors as was indicated by the tests performed. In tests *3* and *4, neural networks 1* and *4* were tested with coarse-coded patterns for generalization on unseen test patterns, after completing the training with a training set. The results are as shown in *tables 8* and *9*.

5. Conclusions

We have developed and tested a connectionist non-monotonic reasoning system using distributed coarse-coded representations. The system has successfully performed the non-monotonic reasoning tasks. The system has displayed good generalization ability on unseen test patterns. The coarse-coded reasoning system exhibited high fault tolerance under noise conditions. These artificially introduced errors were simulating noise conditions.

References

[1] Lokendra Shastri, "Advances in SHRUTI: a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony", Applied Intelligence, Vol 11, No. 1, pp. 79-108, 1999.

[2] Lokendra Shastri, and C. Wendelken, "Multiple instantiation and rule mediation in SHRUTI", Connection Science, Vol 16, pp. 211-217, 2004.

[3] Lokendra Shastri, and C. Wendelken, "Seeking coherent explanations -- a fusion of structured connectionism, temporal synchrony and evidential reasoning", Proceeding of Cognitive Science2000 Conference, Philadelphia, August 2000.

[4] Ron Sun, "On variable binding in connectionist networks", Connection Science, Vol 4, pp. 93-124, 1992.

[5] Sriram. G. Sanjeevi, and P. Bhattacharyya, "Connectionist Reasoning System using Coarse-coded Distributed Representations", Proceeding of ICSCI2005 Conference, Hyderabad, pp723-728, January 2005.

[6] Sriram. G. Sanjeevi, and P. Bhattacharyya, "A fault tolerant Connectionist Model for Predicate Logic Reasoning, Variable Binding: using Coarse-coded Distributed Representations", WSEAS Transactions on Systems, Volume 4, No. 4, pp. 331-336, Apr. 2005.

[7] Sriram. G. Sanjeevi, and P. Bhattacharyya, "A Connectionist Model for Predicate Logic Reasoning using Coarse-coded Distributed Representations", Proceeding of KES2005 Conference, Melbourne, pp. 732-738, September 2005.

[8] Stuart Russell, and P. Norvig, Artificial Intelligence A Modern Approach, Pearson Education, Delhi, 2003.

[9] Geoffrey. E. Hinton, James. L. McClelland, and David. E. Rumelhart, Parallel Distributed Processing, Vol 1, MIT Press, Cambridge, Massachusetts, 1986.

[10] Simon Haykins, Neural Networks, Pearson Education, Delhi, 2001.