ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

Connectionist Rule Processing Using Recursive Auto-Associative Memory

Michael St Aubyn and Neil Davey

Faculty of Information Sciences, University of Hertfordshire
Hatfield, Herts. AL10 9AB. England.

A limitation of many rule-based connectionist models is their dependence on structure to explicitly represent rules, and their consequent inflexibility in acquiring and applying novel rules. A model is described in which recursive auto-associative memory (RAAM) is used as an encoding mechanism to prepare rules of variable structure and content for input to a connectionist rule applicator. The encoded rules are applied 'holistically' to perform simple operations upon binary strings.

1. Introduction

The implementation of rule processing in neural networks has been investigated both as a means of improving the performance of 'classical' rule-based systems, and to model certain aspects of human cognitive performance. Typically, such implementations involve an explicit representation of the rules within a neural network's architecture, exploiting the correspondence between the compositional symbol structure of a rule and the patterns and interactions of nodes and links that can be embodied in a connectionist model (e.g., Shastri & Ajjanagadde 1993).

A weakness of this approach is the inflexibility of structure-based representations and the consequent difficulty of inducing change (and therefore learning) within them. At best, their capacity to adapt and learn is confined to the adjustment of link weights within a predefined architecture.

In this paper, a different approach is described, one in which rules are encoded using Recursive Auto-Associative Memory (RAAM), then processed 'holistically' in a connectionist rule applicator. This approach potentially frees the rule model from some of the limitations inherent in the use of structure-based representations.

2. Recursive Auto-Associative Memory

RAAM was devised by Jordan Pollack (1988) as a means of presenting variable-sized sentences to connectionist processing systems. The RAAM architecture takes as input an arbitrary (tree-like) symbolic structure to be represented and returns as output a fixed-width distributed representation suitable for input to some other connectionist

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

model such as a backpropagation net. In this sense RAAM may be seen as an encoding mechanism, the pre-processing component of some larger system. RAAM also provides the decoding mechanism to convert distributed representations back into structured objects. A RAAM-based system might therefore have a three-level architecture (e.g., Chalmers 1990), with RAAM as the outer two levels and some other processing component in the middle, operating *holistically* upon the implicitly-structured representations which are then RAAM-decoded back into symbol structures. Other RAAM-based models have been used to investigate the accessibility of the implicitly-encoded features of a symbol structure for classification tasks (e.g., Bodén & Niklasson 1995, Sperduti *et al.* 1995).

3. A model of holistic rule application

A model is being developed which exploits this RAAM-encoding mechanism to convert structured rules into a distributed form suitable for connectionist processing. The encoded rules are used to perform simple transformations upon binary data strings. In this model, there is no dependence upon explicit rule-representation by means of dedicated nodes and links. The rules are *implicitly* encoded in distributed representations and *holistically* applied to data strings.

A principal aim of the model is to investigate the extent to which structure-based rule processing is possible without structure-based representation, and to optimise the forms of representation that are most effective for this task.

3.1 The architecture of the model

The model has a two-part architecture, consisting of an *encoding component* and an *application component* (Figure 1). The encoding component is the RAAM. Rules are constructed from a lexicon of symbols using a simple grammar, then recursively encoded by the RAAM into a fixed-width distributed representation. A rule consists of an antecedent (a test) and a consequent (an action). A test, in the current version of the model, examines one bit of the data string to determine its state (on or off). If the test succeeds, an action is triggered, which may result in a change of state of one bit (not necessarily the same one) in the data string.

A rule is applied to a binary data string by the application component, which is a three-layer back-propagation net. The input layer is clamped with a RAAM-encoded reduced description of a rule and a binary representation of a data string. After passing through the hidden layer, the activations in the output nodes are interpreted as a representation of the data string after transformation by the rule.

3.2 Training

The two parts of the model are trained individually. The first objective is to induce

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

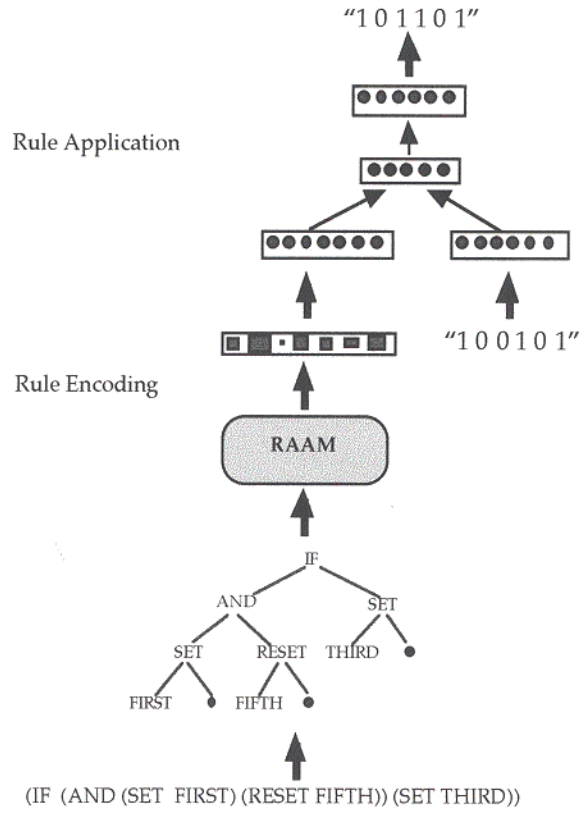

Figure 1. A model of holistic rule application.

learning and generalisation in the RAAM. This is done by creating subsets of the set of all possible rules, and using backpropagation over the recursive auto-associative process to adjust the weights of the RAAM until the model is capable of (a) encoding and decoding all members of the training set of rules, and (b) generalising to the extent that it can also encode and decode, with an acceptable degree of error, the remaining rules from the exhaustive set. Cluster analysis of the reduced descriptions of the rules is used to determine how the RAAM is characterising the data. If the RAAM is using its hidden units to classify the rules according to 'sensible' criteria, it is likely that the reduced descriptions will contain the significant content necessary for rule-application in the second part of the model.

The application component is trained by backpropagation to associate an input, consisting of a reduced description of a rule (produced by the RAAM) plus a binary data string, with the result of applying the rule to the string. Success here is measured in terms of the model's ability to produce correct resultant strings for novel combinations of *rule+data*.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

3.3 Results of RAAM training

In current tests on the model, the following simple rule grammar is used. Note that the conjunction in Figure 1 is not supported by this grammar.

```
Rule -> "IF" Antecedent Consequent
Antecedent -> Test Argument
Consequent -> Operation Argument
Test -> "SET" or "RESET"
Operation -> "SET" or "RESET"
Argument -> "FIRST" or "SECOND" or "THIRD"
```

The grammar will produce an exhaustive set of 36 rules. An example rule is

```
(IF (SET FIRST) (RESET THIRD)),
```

which reads 'If the first bit of the binary string is set to 1, reset the third bit to 0'. (The binary strings for these tests are three bits wide.)

In order to train the RAAM, the rules first undergo a conversion process in which each symbol is replaced by a binary code from a look-up table. The predicates have three-bit codes, the arguments seven-bit codes. The input to the RAAM consists of one predicate plus two arguments plus two one-bit flags (which encode structural information): 19 bits in total. Since the arguments for the next cycle of RAAM encoding are taken from the hidden layer of the RAAM in the previous cycle, the width of the hidden layer must be the same as the size of the argument representation, i.e., seven units. So the architecture of the RAAM is determined as 19-7-19.

Initially, the RAAM was trained on the exhaustive set of rules, the error value settling within 5,000 epochs. After training, the RAAM could encode and decode all rules in the set with 100% accuracy (using a best-match algorithm to decode the representations at the output layer of the RAAM back into symbols). Cluster analysis was performed on the hidden layer of the RAAM to determine which regularities were being represented. With each test, it was discovered that the RAAM was using its hidden units to divide the rules according to certain criteria, such as presence/absence of 'SET' in antecedent predicate position, presence/absence of 'SET' in consequent predicate position, and presence/absence of 'FIRST' in antecedent argument position. These divisions were adhered to without exception, indicating that the RAAM was not only representing the rules in the hidden layer reduced descriptions, but doing so in a very precise and systematic way.

RAAM generalisation was tested by removing groups of rules (selected at random) from the exhaustive set and training the RAAM on the remainder. After training on half of the rules, the RAAM could encode and decode the remaining half with a success of 96.2% (this is the percentage of correctly decoded and situated symbols in the rules of the testing sets over 40 trials, each trial involving a different random selection of rules in both sets).

The results of RAAM training, albeit with a very simple rule-base, show that generalisation is possible within the rule encoder, the analysis revealing that this

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

is being achieved through the categorisation of the data according to significant regularities within the rule-base. The fact that the RAAM is extracting these regularities and representing them in the reduced descriptions generated in its hidden layer indicates that this information will be available to (and hopefully accessible by) the application component.

3.4 Results of rule application

The rule applicator has been implemented and trained on reduced descriptions of rules generated by the RAAM, these being combined with binary data strings at the input of a three-layer backpropagation net which is trained to produce the result of rule application on its output layer (Figure 1). The input layer of the application net is 10 units wide (seven for the reduced description of the rule, plus three for the binary data), the output layer is three units (the resultant) and the hidden layer has been chosen through experimentation to be 12 units. Since the binary strings are three bits wide and given that there are 36 possible rules, there are 288 (8*36) possible combinations of *rule+data* - 72 of which result in a change in the data string.

The applicator was trained and tested using subsets of these 288 combinations. In order to compensate for the fact that in most cases (216 out of 288, or three-quarters) there is no change in the data after rule application (which creates a strong bias for the applicator simply to reproduce the same data string on the output layer), the following method was used to construct the subsets. The set of 72 *rule+data* combinations which *do* produce a change in the data were added to another set of 72 combinations selected at random from the remaining 216. This produces a set of 144 combinations from which are constructed a training set and a test set. The rule representations in all trials were produced by a RAAM which had been fully trained on the entire set of 36 rules.

In the initial trials, the applicator was trained for 6,000 epochs on a set of 72 rules (randomly selected for each trial using the above criteria) and then tested on a set of 72 different rules. Over 40 such trials, the trained applicator could produce the correct resultant data string in 100% of training set cases and 60.8% of test set cases (after the values on the output layer had been rounded to the nearest integer values). Looking just at the test set instances of *rule+data* which should produce a transformation in the data string, the success rate over the 40 trials was 56.6%. In other words, the applicator could correctly perform tests and transformations involving novel combinations of rule and data in just over half the cases. In almost all cases where it failed to produce an expected transformation, the original data was recreated on the output layer. However, an examination of the actual output values in these cases usually revealed a noticeable 'drift' towards the desired result.

Performance of the applicator was improved by increasing the size of the training set relative to the test set. With 40 trials involving 108 combinations in the training set and 36 in the test set, performance in the test set increased to 66.7%, and 63.9% among those cases which should produce transformations. The best results were obtained in a series of 40 trials with sets of 142 and two, where the success rate

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 225-230

in the two-member test sets was 84.2%, and 72.5% among the test cases which should produce transformations in the data strings.

4. Conclusions and future work

The results obtained with the rule encoder indicate that the RAAM is capable of generating reduced descriptions which preserve, in implicit form, the content of the rules that it has been trained on, and that it can learn to encode novel rules from the same grammar with a similarly high degree of recoverability (96.2%). Furthermore, the experiments with the rule applicator indicate that this implicit content is accessible to the extent where it may be used to control simple one-bit tests and actions on data strings presented to a connectionist network. The generalisation tests with the applicator suggest that the model is learning *how* to apply the rules - in at least half the tests involving novel combinations of *rule+data* the rule applicator was able to produce an appropriately modified data string, the success rate rising to 72.5% in the best case.

The grammar and corresponding rule-base in the experiments conducted so far are very simple. The intention is to advance the complexity of the model, and the rules that it can handle, along various dimensions. This will include the addition of rules that involve conjunctive components (e.g., the rule in Figure 1) and a gradual expansion of the lexicon to allow the rules to be applied to larger data strings and to perform a wider range of tests and actions over those strings.

References

Bodén, Mikael & Niklasson, Lars (1995) 'Features of Distributed Representations for Tree-structures: A Study of RAAM' in *Proceedings of the 1995 Swedish Conference on Connectionism*, pp. 121-139.

Chalmers, David J. (1990) 'Syntactic transformations on distributed representations' in *Connection Science*, Vol.2, Nos 1 & 2.

Pollack, J. B. (1988) 'Recursive auto-associative memory: devising compositional distributed representations', *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.*

Shastri, L. & Ajjanagadde, V. G. (1993), 'From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables and Dynamic Bindings Using Temporal Synchrony', *Behavioral and Brain Sciences*, 16, pp. 417-494.

Sperduti, A., Starita, A. & Goller, C. (1995), 'Learning Distributed Representations for the Classification of Terms', *IJCAI 95*, Vol.1, pp. 509-515.