

Distributed Patterns as Hierarchical Structures

Stan C. Kwasny
Barry L. Kalman
Nancy Chang †

Department of Computer Science
Washington University, Campus Box 1045
St. Louis, Missouri 63130

ABSTRACT

Recursive Auto-Associative Memory (RAAM) structures show promise as a general representation vehicle that uses distributed patterns. However training is often difficult, which explains, at least in part, why only relatively small networks have been studied. We show a technique for transforming any collection of hierarchical structures into a set of training patterns for a sequential RAAM which can be effectively trained using a simple (Elman-style) recurrent network. Training produces a set of distributed patterns corresponding to the structures.

1. Introduction

Hierarchical structures have a variety of uses in symbolic computing as representations that can be composed, modified, and otherwise manipulated. Connectionist models have considerable difficulty representing and manipulating such structures within fixed-length vectors with limited computational power. For rapid progress to be made in developing representation schemes for neural networks, combined symbolic and sub-symbolic approaches show the most promise. Among sub-symbolic representation schemes, perhaps the most general is Pollack's (Pollack, 1990) Recursive Auto-Associative Memory (RAAM), which develops distributed patterns over a fixed-size set of units to represent symbolic structures of fixed valence. The symbolic structures must be known in advance, of course, but the representation scheme develops through training.

RAAMs have been the focus of many studies. In his original studies, Pollack (1989; 1990), demonstrated feasibility as well as limited generalization capabilities in representing a small collection of syntactic parse trees. Chalmers (1990) demonstrated that RAAMs could be fruitfully viewed holistically. With a RAAM as input, he trained a network to extract various pieces of the implicit structure. In a series of studies, Blank, et al (1992), investigated holistic properties of RAAMs and the use of sequential RAAMs to process sequences of words as sentences.

Training, however, can be difficult. Consequently, the practicality of RAAMs is greatly limited. This fact may explain why only moderate size networks have appeared in the literature. According to Pollack (personal communication), progress is being made on the training issue. In this paper, we examine a simplified RAAM variant capable of developing distributed representations for structures of arbitrary valence, but without some of the usual training difficulties. The training problem is reduced to that required for a simple recurrent network (SRN) as described by Elman (1990).

We argue that a method based on sequential RAAMs is preferable and present an effective method for training sequential RAAMs which has been successfully applied to a large collection of tree structures of various sizes. Performance is judged based on the ability of the RAAM network to encode and decode structures effectively for increasingly large structures. We rate the sequential RAAM performance as remarkably good.

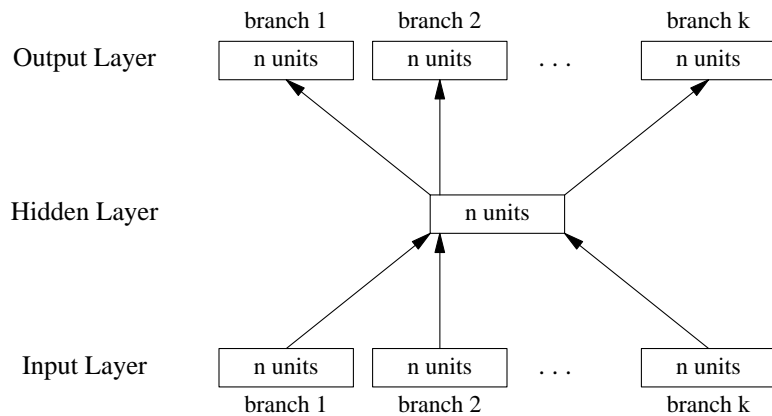
2. RAAM Architecture

The design of the General RAAM architecture is shown in Figure 1. Training is designed to approximate an identity mapping from inputs to outputs and is hence auto-associative. Patterns that develop at the hidden layer

This material is based upon work supported by the National Science Foundation under Grant No. IRI-9201987.

† Supported under National Science Foundation Grant No. CDA-9123643.

Figure 1: General RAAM Architecture
(Based on similar figure in Blank, et al, 1992)



represent a compression of the k branches that have been presented. By examining the network one weight-layer at a time, the functionality can be better understood. The encoding part of the network is represented by the weights connecting input layer to hidden layer. When these weights are applied, a compressed representation results at the hidden layer which when acted upon by the weights connecting hidden to output layer produces the k branches once more. Each branch may be a RAAM itself, the result of applying the encoding circuitry iteratively, thus allowing the representation of nested structures. The decoding circuitry is applied iteratively to recover the structure.

Note that the valence of the structure must be known in advance when the network is constructed. Further note that whether a branch is a RAAM or a terminal symbol with some chosen representation, it requires n units to represent it.

3. Symbolic Mappings

Hierarchical structures, in our approach, are considered to be forests of ordered trees. The nodes of the trees must contain symbols from a finite alphabet and each symbol is assigned a representation.

Each forest can be mapped to a binary tree, which in turn can be mapped to a simple sequence of symbols in which empty subtrees are marked (Knuth, 1973). Such a mapping is invertible so that for every hierarchical structure there is an ordered sequence of symbols and vice versa. This reduces the general RAAM training problem for arbitrary hierarchical structures to a sequential RAAM training problem, which is more manageable.

Figure 2 shows the series of steps involved for a sample tree. A general parse tree is transformed into a binary tree which is rendered as an ordered list of symbols which, in turn, can be used to produce training sequences for the SRN. Once training is complete, the RAAM encoding and decoding subnetworks can be used iteratively to produce a representation for any sequence of symbols from the alphabet.

For our purposes, we use hierarchical structures that arise as intermediate and final results during parsing and intend to use the RAAM representations as targets for a parser under development (see, Kwasny & Faisal, 1992; Kwasny & Kalman, 1992).

4. Recurrent Training

Only a slight variation of the training methods used for the SRN is required. In preparing the input data, the beginning of each new sequence is marked. This tells training that the activation pattern from the hidden layer in the previous step is not to be copied to that part of the input layer, but that the activation of those input units is the empty pattern. For sequential RAAMs, we additionally need to specify that training is auto-associative and to omit the target portion of each pattern.

Figure 2: Transformation from Hierarchical Structure to Binary Tree to Sequential List to RAAM

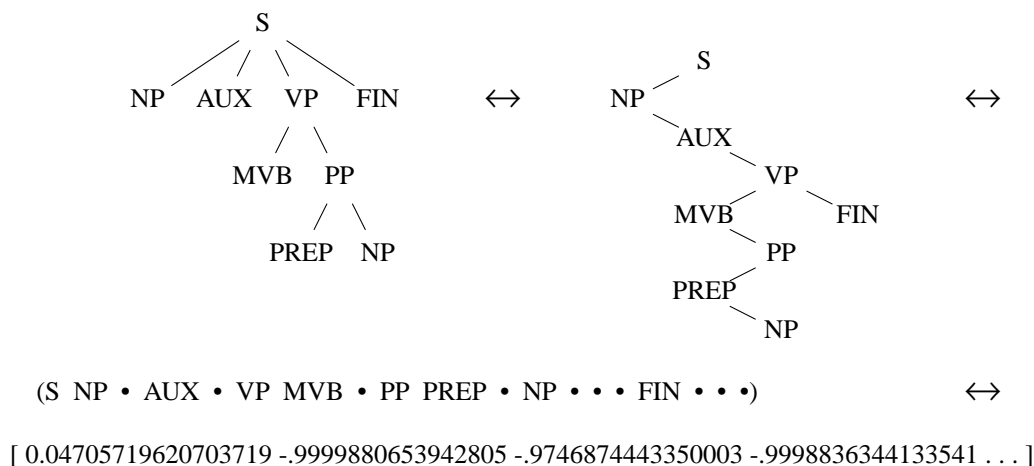
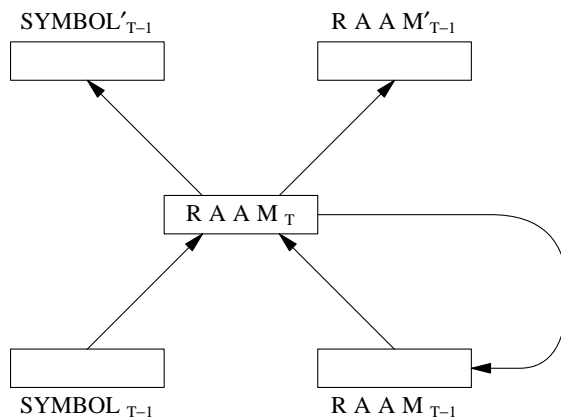


Figure 3 shows how training proceeds. On each step, SYMBOL_{T-1} receives its value from the next symbol in the sequential list. The RAAM_{T-1} portion is initially empty (a chosen pattern representing the empty structure) and on each subsequent step in the sequence its value is the pattern present on the hidden layer of the previous step. The output layer is likewise divided into two segments and values approximating the corresponding input values appear as a result of the mapping. Training of the network reinforces its ability to compress the previous distributed RAAM representation and the current symbol. The two weight layers are trained to work in unison with the decoding part of the network approximating the inverse of the encoding part.

Contrary to RAAM training described in the literature, stratified or staged training schedules become unnecessary with this type of training. The architecture of the network does not reflect in any way the structure of the particular hierarchical structure being encoded. Structures are encoded from the sequence left-to-right (analogous to top-down in the tree) and decoding produces structures right-to-left (analogous to bottom-up in the tree).

Training sequential RAAMs differs from training ordinary SRNs. From Figure 3, we see that for each unit k associated with the RAAM portion of the output, $(\text{RAAM}'_{(T-1)k} - \text{RAAM}_{(T-1)k})$ is contributed to the error function. Since the activations of these units depends on network parameters, a correction term must be added to the gradient computation associated with the connection from RAAM_{T-1} to RAAM_T . In our experience, without this correction term, calculations during training contain sufficient error to adversely affect its success.

Figure 3: Sequential RAAM Training



5. Results

For this experiment, we examined a grammar containing 23 rules and derived 25 different syntactic parse trees that the grammar was capable of producing on steps leading to a parse. These served as training examples for the sequential RAAM which contained 21 units. The symbol portion of the network also required 21 units. One of these trees is shown in the example in Figure 2. Following training, we tested the performance of the iterated encoding and decoding process by testing with these same structures. While training and testing with identical data is not a valid evaluation of the training, in this case it does show the effectiveness of the encoding/decoding subnetworks when iteratively applied. Training only attempts to perfect the individual single-step operation. The real effectiveness of the method lies in its ability to repeatedly and consistently operate on the representation.

In the course of developing representations for the 25 unique sequences of symbols, a total of 75 unique RAAMs were developed, including the empty RAAM. These included representations for sequences of lengths 0 through 15. Training produced only one small structural error in one of the 25 sentences, which led to redundancy in one part of the sequence. There also were 4 symbolic errors out of a possible 222 symbols tested for decoding, but these were uniformly minor, differing in one unit position.

Hierarchical clustering analysis was used to determine general strategies used in the representation for both the 75 RAAMs and the 25 unique sequences. In studying these results, two strategies seem to be predominant. First, clustering was based on length. In part, this may relate to the training strategy we used which trained shorter segments first, adding the longer ones later. Second, clustering was based on the most recently added symbols. This is easy to see since the decoding subnetwork must be able to reliably extract the most recent symbols, leaving a RAAM to be further decoded.

6. Summary and Conclusions

While RAAMs look very promising as fixed-size representations of hierarchical structures, training is difficult and the network architecture demands that the valence of the hierarchy be known when the network is created. Using sequential RAAMs and our slightly modified SRN training algorithm, unlimited structures may be attained. While training is still difficult, there are good strategies for training with one-dimensional input data. For example, short sequences can be used to train the network prior to training longer sequences. In practice, this strategy produced weights with the best overall performance in this study.

References

- Blank, D.S., Meeden, L.A., and Marshall, J.B. (1992). Exploring the Symbolic/Subsymbolic continuum: A Case Study of RAAM. In J. Dinsmore (ed), *Closing the Gap: Symbolism vs. Connectionism*. Lawrence Erlbaum Associates, pp. 113–148.
- Chalmers, D.J. (1990). Transformations on Distributed Representations. In Noel Sharkey (ed), *Connectionist Natural Language Processing*. Intellect Publishers, Oxford, pp. 46–55.
- Elman, Jeffrey L. (1990). Finding Structure in Time. *Cognitive Science*, 14, 179–212.
- Kalman, B.L. (1990). Super Linear Learning in Back Propagation Neural Nets. Technical Report WUCS-90-21, St. Louis: Department of Computer Science, Washington University.
- Knuth, Donald E. (1973). *The Art of Computer Programming*. Volume 1: Fundamental Algorithms. Addison-Wesley, Reading MA.
- Kwasny, Stan C., and Kalman, Barry L. (1992). A Recurrent Deterministic Parser. *Proceedings of the Fourth Midwest Artificial Intelligence and Cognitive Science Society Conference*, Intergraph Corporation, Huntsville, Alabama, 82–86.
- Kwasny, Stan C., and Faisal, Kanaan A. (1992). Symbolic Parsing Via Subsymbolic Rules. In J. Dinsmore (ed), *Closing the Gap: Symbolism vs. Connectionism*. Lawrence Erlbaum Associates, pp. 209–235.
- Pollack, Jordan. (1989). Implications of Recursive Distributed Representations. In David S. Touretzky (ed), *Advances in Neural Information Processing Systems*. Morgan Kaufmann, Los Gatos, CA.
- Pollack, Jordan. (1990). Recursive Distributed Representations. *Artificial Intelligence*. Vol. 46, pp. 77–105.
- Stolcke, Andreas, and Wu, Dekai. (April, 1992). Tree Matching with Recursive Distributed Representations. TR-92-025, International Computer Science Institute, Berkeley, CA.

