Learning Distributed Representations for the Classification of Terms*

Alessandro Sperduti, Antonina Starita
University of Pisa
Dipartimento di Informatica
Corso Italia 40, 56125 PISA
Italy

Christoph Goller
Institut fiir Informatik
Technische Universitat Mimchen
D-80290 Miinchen
Germany

Abstract

This paper is a study on LRAAM-based (Labeling Recursive Auto-Associative Memory) classification of symbolic recursive structures encoding terms. The results reported here have been obtained by combining an LRAAM network with an analog perceptron. The approach used was to interleave the development of representations (unsupervised learning of the LRAAM) with the learning of the classification task. In this way, the representations are optimized with respect to the classification task. The intended applications of the approach described in this paper are hybrid (symbolic/connectionist) systems, where the connectionist part has to solve logic-oriented inductive learning tasks similar to the term-classification problems used in our experiments. These problems range from the detection of a specific subterm to the satisfaction of a specific unification pattern, and they can get a very satisfactory solution by our approach.

1 Introduction

In this paper, we show that basic classification tasks on symbolic recursive structures (logical terms) may be solved with a connectionist approach. We regard this approach to logic-oriented inductive learning as a supplement to classical symbolic AI, which is mainly deductive. It could be used in hybrid (symbolic/connectionist) systems, especially for lifting the expressiveness of these systems from prepositional logic to predicate logic.

The generalizations that are necessary for the term-classification problems presented in this paper range from the detection of specific subterms to finding a most specific term, subsuming all terms from the positive class. While these tasks, if known in advance, are trivially solved by ad hoc symbolic procedures, it is very difficult to induce their nature when only a set of positive and negative examples is given. The problem of representing recursive structures of arbitrary size has been so far the major obstacle to using connectionist approaches for such logic-oriented inductive learning tasks. Neural networks like the LRAAM-model (Labeling Recursive Auto-Associative Memory) [Sperduti, 1993a; 1993b], however, propose a solution to this problem. The LRAAM is an unsupervised method for devising fixed-width distributed representations of labeled variable-sized recursive data structures, such as graphs, lists and logical terms. The appropriateness of these distributed representations for subsequent classification tasks, in the context of natural language processing, has been shown e.g. in [Cadoret, 1994], where the distributed representations of syntactical trees devised by an LRAAM are automatically classified according to the typology of dialogue acts. Our approach, however, is a little bit different. We interleave the unsupervised development of representations (learning of the LRAAM) with the supervised learning of the classification task. In this way the representations are optimized with respect to the classification task. The classification becomes much easier. In fact, in all the problems we investigated, we did not need an additional complex multilayer network for classification. A single sigmoidal unit connected to the hidden layer of the LRAAM is sufficient.

In section 2 we introduce the LRAAM-model. In section 3 we describe the different options for using the LRAAM for classification tasks. The approach chosen by us is introduced within this framework. In section 4 the term-classification tasks used for our experiments are described. In section 5 we present the results and an analysis of the representations found for the terms. In Section 6 we conclude proposing directions for future research.

2 Labeling RAAM

The *Labeling RAAM (LRAAM)* [Sperduti and Starita, 1993; Sperduti, 1993a; 1993b; 1994] is an extension of the RAAM model [Pollack, 1990] which allows one to encode labeled structures. The general structure of the network for an LRAAM is shown in Figure 1. The network is trained by backpropagation^1 to learn the identity

*This work *was* partially supported by the DFG, EC, DA AD and CRUI (Vigoni Project).

^{1}We verified faster learning using the *descending-epsilon* heuristic technique [Yu and Simmons, 1990]: during the learning phase we maintain a list of the patterns having a decoding error higher than a specified threshold. The back-

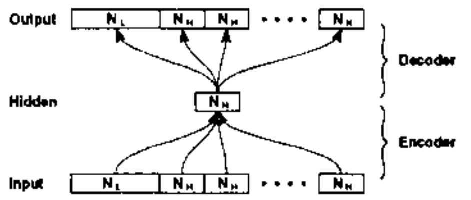

Figure 1: The network for a general Labeling RAAM.

function. The idea is to obtain a compressed representation (hidden layer activation) of a node of a labeled directed graph by allocating a part of the input (output) of the network to represent the label (*NL* units) and the rest to represent one or more pointers. This representation is then used as pointer to the node. To allow the recursive use of these compressed representations, the part of the input (output) layer which represents a pointer must be of the same dimension as the hidden layer (*NH* units). Thus, a general LRAAM is implemented by a $N_I - N_H - N_I$ feed-forward network, where $N_I = N_L + nN_H$, and n is the number of pointer fields.

Labeled directed graphs can be easily encoded using an LRAAM. Each node of the graph only needs to be represented as a record, with one field for the label and one field for each pointer to a connected node. The pointers only need to be logical pointers, since their actual values will be the patterns of hidden activation of the network. A graph is represented by a list of these records, and this list constitutes the initial training set for the LRAAM. During training the representations of the pointers are consistently updated according to the hidden activations. Consequently, the training set is dynamic. In the original formulation of the LRAAM, at the beginning of training the representations for the non-void pointers and void pointers are set at random. After each epoch, depending on the hidden activation obtained in the previous epoch for each pattern, the representations for the non-void pointers in the training set are updated. The void representations are, on the other hand, copied from the output.

The convention used for the void pointers, however, has the drawback that no fixed representation for the void pointer is used. This means that when we want to encode a structure not in the training set we do not know which representation to use for the void pointer. One simple solution to this problem was proposed by Cadoret [Cadoret, 1994]: the void pointer is represented by a vector with null components and its output error not considered when backpropagating the error across the network. In this paper, we adopted the same strategy.

Once the training is complete, the patterns of activation representing pointers can be decoded to retrieve information. In order to decide whether a pointer is void

propagation is performed only on the patterns of the list: when all patterns are below the threshold, we lower it and resume the backpropagation. The procedure stops when we obtain the perfect decoding.

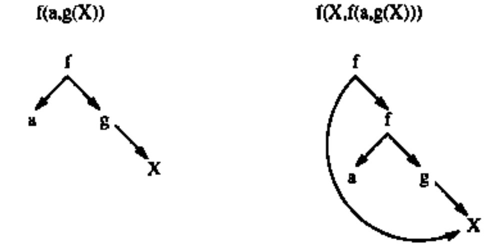

Figure 2: Example of terms represented as LDAGs.

or not, one bit of the label is allocated for each pointer field to represent the void condition. Notice that multiple labeled directed graphs can be encoded in the same LRAAM.

Encoding of Terms

For our purpose, logical terms can conveniently be represented as labeled directed acyclic graphs (LDAGs), where function symbols are mapped to internal nodes, while constants and variables are mapped to terminal nodes. The label of each node is used to store the symbol associated to the assigned entity. Some examples of terms represented by LDAGs are given in Figure 2. The advantage of this representation consists in the possibility to uniquely represent identical subterms within a term, as shown on the right side of Figure 2, where the same variable *X* appears both as first argument of the function *f(,)* and as argument of the function *g()*. This feature allows us to represent terms very compactly. When considering a set of terms, we use the same representational strategy used for a single term: each term (or subterm) is represented only once , and repetitions of the same term are handled by resorting to pointers. An example is given in Figure 3. Notice that, an LRAAM trained to encode a set of terms will generate a reduced representation for each intermediate term representation (pointer to a subtree) regardless of the fact that it constitutes a term we are interested in or not.

In this paper, we will consider only the representation and classification of *ground* terms, i.e., terms which do

Notice that the space and time complexity of the learning algorithm depends on the number of subterms in the training set.

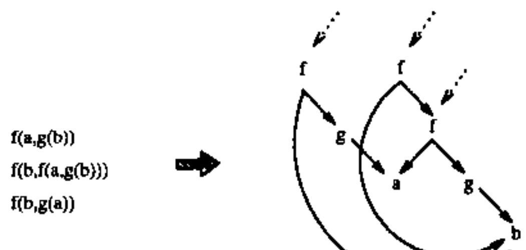

Figure 3: Representation of a set of terms.

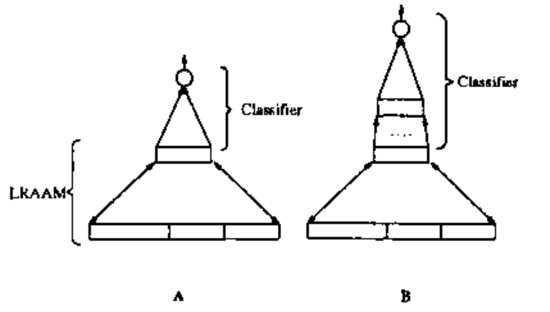

Figure 4: Two different networks for the classification of reduced descriptors devised by an LRAAM.

not involve variables, however, the classification tasks we propose involve the concept of logical variable.

3 Classification of Terms

The main subject of this report is to demonstrate the feasibility of classification of ground terms encoded by an LRAAM. The aim is to devise a system able to generalize on a wide range of classification tasks, from the detection of a particular symbol within the term to be classified to the recognition that the term satisfies a given unification scheme. For this, we propose to classify the reduced representations devised by an LRAAM through a feed-forward neural network. Specifically, we propose to use one of the network architectures shown in Figure 4. In both of them, the first, part is constituted by an LRAAM (notice the double side arrow connections) whose task is to encode the terms as discussed in the previous section. The classification task is then performed in the second part of the network through a simple sigmoidal neuron (network A) or a multi-layer feed-forward network with one or more hidden layers (network B). A very similar approach was used by Stolcke and Wu in [Stolcke and Wu, 1992], where they try to learn how to unify very simple terms encoded in a RAAM.

Several options for the training of the proposed architectures are available. First of all note that, given a set of positive and negative terms, we have no information regarding the classification of subterms, which can also appear within both positive and negative terms. This means that the classifier will be used only on reduced representations of terms and its output will be disregarded on reduced descriptors of subterms.

Assuming that the training of the system may end when the classification task is performed correctly, the different options we have for the training can be characterized by the proportion of the two different learning rates (for the classification error and the decoding error) and by the different degrees x,y of presence (or absence) of the following two basic features:

- the training of the classifier is started not until x percent of the training set is correctly encoded and successively decoded by the LRAAM;
- the error coming from the classifier is backpropagated across y levels of the structures encoded by the LRAAM[3].

Notice that, even if the training in the classifier is started only when all the structures in the training set are properly encoded and decoded, still the classifier's error can change the reduced representations which, however, are maintained consistent by learning in the LRAAM.

The reason for allowing different degrees of interaction between the classification and the representation tasks may be due to the necessity of having different degrees of adaptation of the reduced representations to the requirements of the classification task. If no interaction at all is allowed, i.e., the LRAAM is trained first and then its weights frozen (y= 0), the reduced representations will be such that similar representations will correspond to similar structures, while if full interaction is allowed, i.e., the LRAAM and the classifier are trained simultaneously, the reduced representations will be such that structures in the same class will get very similar representations[4].

In this paper, we explore the classification capabilities of the network architecture A, i.e., a single sigmoidal unit connected to the hidden layer of the LRAAM. The training algorithm we used is as follows:

- the training of the classifier is started simultaneously (x= 0) with the training of the LRAAM;
- the error coming from the classifier is backpropagated only to the pointers of the terms to be classified (y= 1).

We will see that, even with this very simple architecture and learning algorithm, we are able to obtain very good results.

4 Description of the Classification Problems

In order to test the ability of the proposed architecture to deal with several classification tasks involving terms, we have generated a set of "simple" classification problems. We have summarized the characteristics of each problem in Table 1. The first column of the table reports the name of the problem, the second one the set of symbols (with associated arity) compounding the terms, the third column shows the rule(s) used to generate the positive examples of the problem[0], the fourth column reports the number of terms in the training and test set respectively, the fifth column the number of subterms in the training and test set, and the last column the maximum depth[6] of terms in the training and test set.

[3] The backpropagation of the error across several levels of the structures can be implemented by unfolding the encoder of the LRAAM (the set of weights from the input to the hidden layer) according to the topology of the structures.

[4] Moreover, in this case, there is no guarantee that the LRAAM will he able to encode and decode consistently all the structures in the training set, since the training is stopped when the classification task is performed correctly.

[5]Remember that the terms are all ground.

[6] We define the depth of a term as the maximum number of edges between the root and leaf nodes in the term's LDAG-representation.

Classification Problems

Problem	Symbols	Positive Examples.	#terms (tr.,test)	#subterms (tr.,test)	depth (pos.,neg.)
lblocc1 long	f/2 i/1 a/0 b/0 c/0	no occurrence of label c	(259,141)	(444,301)	(5,5)
termocc1	f/2 i/1 a/0 b/0 c/0	the (sub)terms i(a) or f(b,c) occur somewhere	(173,70)	(179,79)	(2,2)
inst1	f/2 a/0 b/0 c/0	instances of f(X,X)	(200,83)	(235,118)	(3,2)
inst1 long	f/2 a/0 b/0 c/0	instances of f(X,X)	(202,98)	(403,204)	(6,6)
inst4	f/2 a/0 b/0 c/0	instances of f(X,f(a,Y))	(175,80)	(179,97)	(3,2)
inst4 long	f/2 a/0 b/0 c/0	instances of f(X,f(a,Y))	(290,110)	(499,245)	(7,6)
inst7	t/3 f/2 g/2 i/1 j/1 a/0 b/0 c/0 d/0	instances of t(i(X),g(X,b),b)	(191,109)	(1001,555)	(6,6)

Table 1: Description of a set of classification problems involving logic terms.

For each problem about the same number of positive and negative examples is given. Both positive and negative examples have been generated randomly. Training and test sets are disjoint and have been generated by the same algorithm.

It must be noted that the set of proposed problems range from the detection of a particular atom (label) in a term to the satisfaction of a specific unification pattern. Specifically, in the unification patterns for the problems instl, instl .long and inst 7 the variable *X* occurs twice making these problems much more difficult than inst 4, because any classifier for these problems would have to compare arbitrary subterms corresponding to *X*. However concerning the data sets presented here, this is true only for instl and instl -long, and it will be shown later, that both of them can be solved quite well by our approach. For problem inst7, randomly generating negative examples leads to terms not very similar to *t(i(X)}g(X,b),b)*. The reason for this is that there are so many possible symbols and that *t(i(X), g(X,* 6), *b)* has already a very specific structure. Therefore there is no negative example being an instance of *t(i(X), g(Y,* 6), 6) with *X* *Y* neither in the training nor in the test set, what may make inst7 unexpectedly easy.

5 Results

In Table 2, we have reported the best result we obtained for each problem, described in Table 1, over 4 different network settings (both in number of hidden units for the LRAAM and learning parameters). The simulations were stopped after 30,000 epochs, apart for problem instl -long for which we used a bound of 80,000 epochs, or when the classification problem over the training set was completely solved. We made no extended effort for optimizing the size of the network and the learning parameters, thus it should be possible to improve on the reported results. The first column of the table shows the name of the problem, the second one the number of units used to represent the labels, the third the number of hidden units, the fourth the learning parameters *(N* is the learning parameter for the LRAAM, *e* the learn-

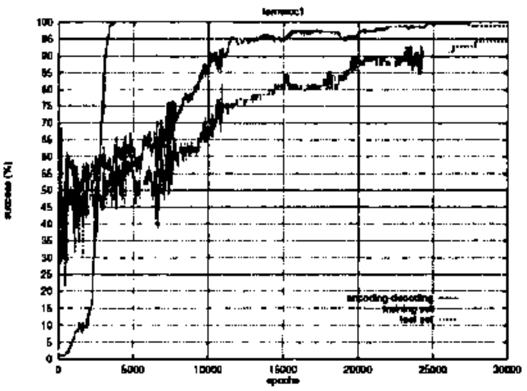

Figure 5: Performance curves for termocc1.

ing parameter for the classifier, u the momentum for the LRAAM), the fifth the percentage of terms in the training set which the LRAAM was able to properly encode and decode, the sixth the percentage of terms in the training set correctly classified, the seventh the percentage of terms in the test set correctly classified, and the eighth the number of epochs the network emploied to reach the reported performances.

From the results, it can be noted that some problems get a very satisfactory solution even if the LRAAM performs poorly. Moreover, this behavior does not seem to be related with the complexity of the classification problem, since both problems involving the simple detection of an atom (label) in the terms (IblocclJLong) and problems involving the satisfaction of a specific unification rule (inst4_long> inst7) can be solved without the need of a fully developed LRAAM. Thus, it is clear that the classification of the terms is exclusively based on the encoding power of the LRAAM's encoder which is shaped both by the LRAAM error and the classification error. However, even if the LRAAM's decoder is not directly involved in the classification task, it helps the classification process since it forces the network to

Best Results

Problem	#L-unit	#H-unit	Learning Par.	% Dec.-Enc.	% Tr.	% Ts.	#epochs
lbloccl long	8	35	$\eta = 0.2,\ \epsilon = 0.001,\ \mu = 0.5$	4.25	100	98.58	11951
termoccl	8	25	$\eta = 0.1,\ \epsilon = 0.1,\ \mu = 0.2$	100	98.84	94.29	27796
inst1	6	35	$\eta = 0.2,\ \epsilon = 0.06,\ \mu = 0.5$	100	97	93.98	10452
inst1 long	6	45	$\eta = 0.2,\ \epsilon = 0.005,\ \mu = 0.5$	36.14	94.55	90.82	80000
inst4	6	35	$\eta = 0.2,\ \epsilon = 0.005,\ \mu = 0.5$	98.86	100	100	1759
inst4 long	6	35	$\eta = 0.2,\ \epsilon = 0.005,\ \mu = 0.5$	8.97	100	100	6993
inst7	13	40	$\eta = 0.1,\ \epsilon = 0.01,\ \mu = 0.2$	1.05	100	100	6158

Table 2: The best results obtained for each classification problem.

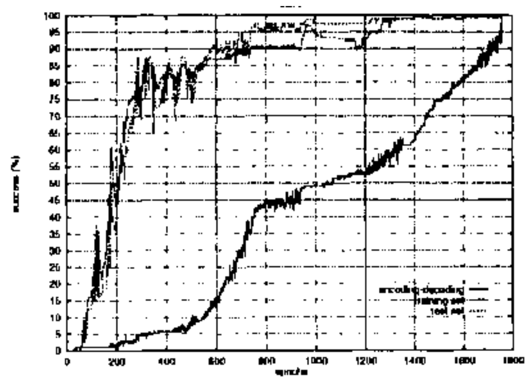

Figure 6: Performance curves for **inst4**.

generate different representations for terms in different classes[7].

In order to give a feeling of how learning proceeds, the performance of the networks during training is shown for the problems termoccl, inst4 and inst4_long in Figures 5-7, where the encoding-decoding performance curve of the LRAAM on the training set is reported, together with the classification curves on the training and test set.

Reduced Representations for Classification

In this section, we briefly discuss the representational differences between a basic LRAAM (without classifier) and the architecture proposed in this paper. The basic LRAAM organizes the representational space in such a way that similar structures get similar reduced representations (see [Sperduti, 1993a] for more details). This happens because, even if the LRAAM is trained in supervised mode both over the output of the network and over the relationships among components (i.e., the information about the pointers), the network is auto-associative and thus it decides by itself the representation for the pointers. Consequently, the learning mode for the LRAAM is, mainly, unsupervised. When a classifier is introduced (as in our system), the training of the LRAAM is no longer mainly unsupervised, since the error of the classifier constrains the learning. The resulting learning regime is somewhat between an unsupervised and a supervised mode.

In order to understand the differences between representations devised by a basic LRAAM and the ones devised in the present paper, we trained a basic LRAAM (of the same size of the LRAAM used by our network) over the training set of instl, then we computed the first, second, and third principal component of the reduced representations obtained both for the training and test set[8]. These principal components are plotted in Figure 8. It can be noted that the obtained representations mainly cluster themselves in specific points of the space. Terms of the same depth constitute a single cluster, and terms of different depth are in different clusters. The same plot for the reduced representations devised by our network (as from Table 2, row 3) is presented in Figure 9. The overall differences with respect to the basic LRAAM plot consists in a concentration of more than half (57%) of the positive examples of the training and test sets in a well defined cluster, while the remaining representations are spread within two main subspaces. The well defined cluster can be understood as the set of representations for which there was no huge interference between the decoder of the LRAAM and the classifier (this allowed the formation of the cluster), while the remaining representations do not preserve the cluster structure since they have to satisfy competitive constraints coming from the classifier and the decoder. Specifically, the classifier tends to cluster the representations into two well defined clusters (one for each class), while the LRAAM decoder tends to develop well distinct reduced representations since they must be decoded to different terms.

The above considerations on the final representations

'Actually, the decoder error forces the LRAAM network to develop a different representation for each term, however, when the error coining from the classifier is very strong, it can happen that terms in the same dass get almost identical representations.

[8]We considered only the reduced representations for terms. No reduced representation for subterms was included in the analysis.

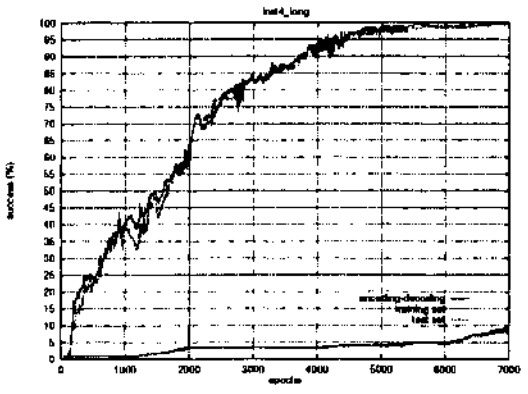

Figure 7: Performance curves for **inst4_long**.

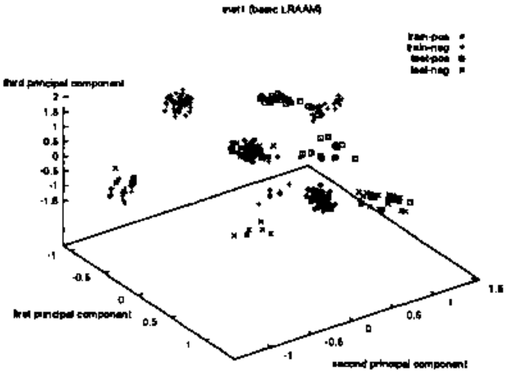

Figure 8: The first, second, and third principal component of the reduced representations, devised by a basic LRAAM on the training and test sets of the **inst1** problem, yield a nice 3D view of the term's representations.

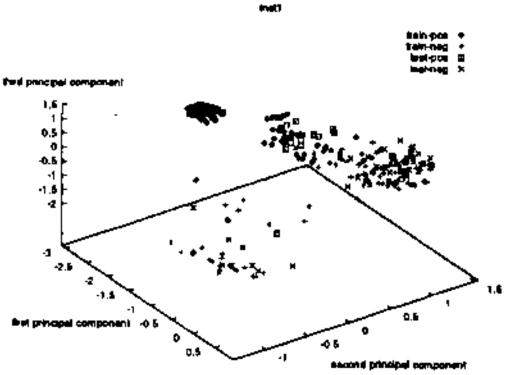

Figure 9: Results of the principal components analysis (first, second, and third principal component) of the reduced representations developed by the proposed network (LRAAM + Classifier) for the **inst1** problem.

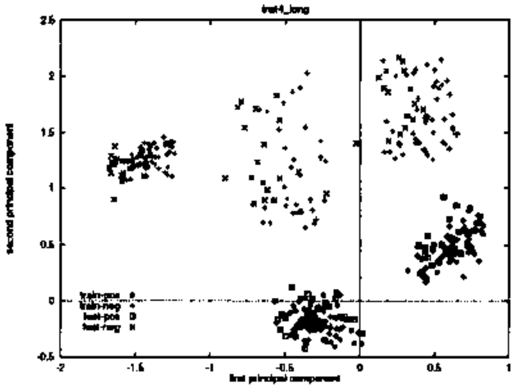

Figure 10: Results of the principal components analysis (first, and second principal component) of the reduced representations developed by the proposed network (LRAAM + Classifier) for the **inst4-long** problem. The resulting representations are clearly linearly separable.

for the terms are valid only if the LRAAM reaches a good encoding-decoding performance on the training set. However, as we have reported in Table 2, some classification problems can be solved even if the LRAAM performs poorly. In this case, the reduced representations contain almost exclusively information about the classification task. In Figure 10 and Figure 11 we have reported the results of a principal components analysis on the representations developed for the problems inst 4 long and inst7, respectively. In the former, the first and second principal components suffice for a correct solution of the classification problem. In the latter, the second principal component alone gives enough information for the solution of the problem. Moreover, notice how the representations developed for inst7 clustered with smaller variance than the representations developed for inst4_long, and how this is in accordance with the better performance in encoding-decoding of the latter than the former. Of course, this does not constitute enough evidence for concluding that the relationship between the variance of the clusters and the performance of the LRAAM is demonstrated. However, it seems to be enough for calling a more accurate study on this issue.

6 Conclusion and Future Work

In this paper, we have proposed a neural network architecture based on the combination of an LRAAM network with an analog perceptron. This architecture can be considered an extension of the SRN by Elman [Elman, 1990]. In fact, when considering our network for the classification of lists, the same architecture as a SRN is obtained, with the difference that there are additional connections from the hidden layer to the input layer[9]. Thus, when considering lists, the only difference between

[9]The output layer of the LRAAM can be considered the same as the input layer.

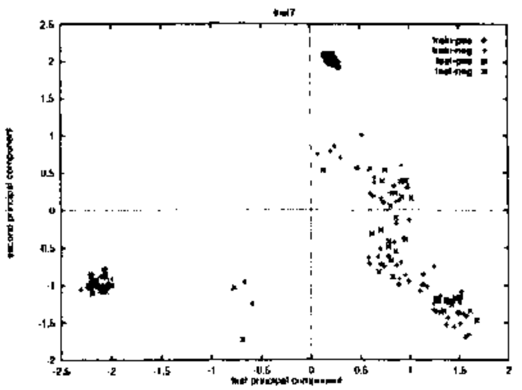

Figure 11: Results of the principal components analysis (first, and second principal component) of the reduced representations developed by the proposed network (LRAAM + Classifier) for the inst7 problem. The resulting representations can be separated using only the second principal component.

a SRN and our network is in the unsupervised learning performed by the LRAAM. However, when forcing the learning parameters for the LRAAM to be null, we obtain the same learning algorithm as in SRN. Consequently, we can claim that SRN is a special case of our network. Moreover, our network can be considered more general with respect to a SRN because it allows the classification of labeled directed graphs.

We have shown that basic classification tasks on complex terms can be solved by our network. With independent test sets we have verified that the networks really find the right generalizations. Based on these very promising results we want to continue our research in the following two directions.

On the one side we want to investigate different network architectures and learning options (see Section 3) in order to achieve faster learning with smaller representations (fewer units in the hidden layer of the LRAAM). Based on the results in this paper, we conclude that it pays to optimize the representations for the classification task. Therefore, the most promising option to investigate seems to be the recursive backpropagation of the classification error over the structures.

On the other side we plan to experiment with more complex examples. The single basic tasks which have been solved separately in this paper (occurrence of a specific label, occurrence of a specific subterm and the satisfaction of a specific unification pattern) have to be combined (disjunctively and conjunctively) to more complex tasks. Furthermore we want to introduce two additional basic tasks: the occurrence of unification patterns within a term and the recognition of simple regular languages on the paths from the top function symbol of a term to its leaves. According to our knowledge, no single symbolic learning system is able to solve all these basic learning tasks or even combinations of them. Our experiments should also be supplemented by examples coming from real applications. The application we are currently working on is a hybrid (symbolic/connectionist) reasoning system. The symbolic component of this system is the theorem prover SETHEO. Connectionist approaches are used within this system to learn search control heuristics from examples [Suttner and Ertel, 1990; Goller, 1994].

References

[Cadoret, 1994] V. Cadoret. Encoding syntactical trees with labelling recursive auto-associative memory. In *Proceedings ECAI,* pages 555-559, 1994. Amsterdam.

[Elman, 1990] J. L. Elman. Finding structure in time. *Cognitive Science,* 14:179-211, 1990.

[Goller, 1994] Christoph Goller. A connectionist control component for the theorem prover setheo. ECAI94 Workshop on Combining Symbolic and Connectionist Processing, 1994.

[Pollack, 1990] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence,* 46(l-2):77—106, 1990.

[Sperduti and Starita, 1993] A. Sperduti and A. Starita. An example of neural code: Neural trees implemented by LRAAMs. In *International Conference on Neural Networks and Genetic Algorithms,* pages 33-39, 1993. Innsbruck.

[Sperduti, 1993a] A. Sperduti. Labeling RAAM. Technical Report 93-029, International Computer Science Institute, 1993. To appear on Connection Science Vol. 6, n. 4, pages 429-459.

[Sperduti, 1993b] A. Sperduti. On some stability properties of the LRAAM model. Technical Report 93-031, International Computer Science Institute, 1993. To appear on IEEE Transactions on Neural Networks.

[Sperduti, 1994] A. Sperduti. Encoding of Labeled Graphs by Labeling RAAM. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6,* pages 1125-1132. San Mateo, CA: Morgan Kaufmann, 1994.

[Stolcke and Wu, 1992] A. Stolcke and D. Wu. Tree matching with recursive distributed representations. Technical Report TR-92-025, International Computer Science Institute, 1992.

[Suttner and Ertel, 1990] C.B. Suttner and W. Ertel. Automatic Acquisition of Search Guiding Heuristics. In *Proceedings of the 10. International Conference on Automated Deduction (CADE),* pages 470-484. Springer LNAI 449, 1990.

[Yu and Simmons, 1990] Y. Yu and R. Simmons. Descending epsilon in backpropagation: a technique for better generalization. In *International Joint Conference on Neural Networks,* pages 167-172, 1990.