T-61.181 Special Course in Information Science I
"Time in Self-Organizing Maps"

SARDNET and SARDSRN

Jussi Ahola

28th October 2002

1. Introduction

This paper is a review of two approaches involving SOM in processing of sequential input data. The first one proposed in the paper by James and Miikkulainen [1] is presented in section 2 and the second one proposed in the papers by Mayberry and Miikkulainen [2], [3] in chapter 3. Since the both approaches are strongly application related, the former to speech processing and the latter to natural language processing, the chapters include also an illustrative example, in addition to the architecture description. Finally, the paper ends up with a discussion about the two approaches.

2. SARDNET

SARDNET (Sequential Activation Retention and Decay NETwork) is a self-organizing neural network designed for sequence classification. The problem involved considers recognition of patterns in a time series of (numerical) vectors, which requires forming a good internal representation for the sequences. The approach taken in SARDNET for solving the problem is to extend the self-organizing map (SOM) with activation retention and decay in order to create unique distributed response patterns for different sequences. As a matter of fact, it uses a subset of SOM nodes to represent the sequence of vectors. More precisely, each sequence is presented by a unique subset nodes, which allows a large amount of sequence representations to be "packed" into a small map. Thus, the network results in a distributed representation of a set of sequences.

2.1. Network architecture

The architecture of the SARDNET is illustrated in Figure 1. The network gets as inputs sequences of n-dimensional vectors $\boldsymbol{S}=\{\boldsymbol{V_1}, \boldsymbol{V_2},..., \boldsymbol{V_l}\}$, where the components of $\boldsymbol{V_i}$ are real values in the interval [0,1]. The network itself consists of an extended version of a standard $m\times m$ SOM. I.e., the nodes of the map are associated not only with an n-dimensional weight vector $\boldsymbol{W_{jk}}$ but also an activation o_{jk}, $1<j,k<m$.

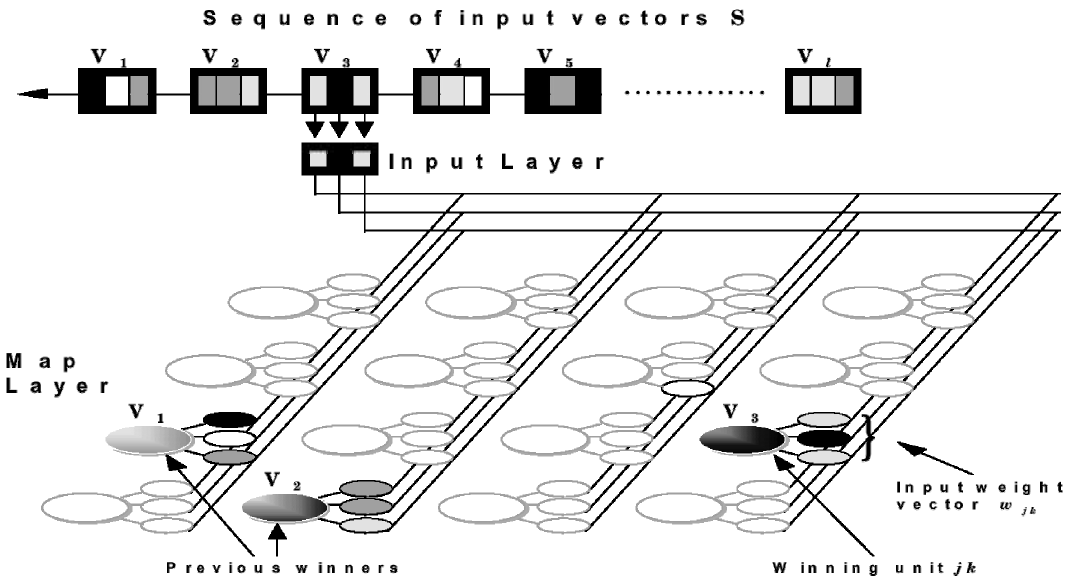


Figure 1 The SARDNET structure.

The training of the SARDNET resembles that of the basic SOM. The major difference is in the winner competition, where in the SARDNET the sequential nature of the inputs is taken into account in the map unit activation. More specifically, once a unit wins an input its activation is set to 1.0 and it is made ineligible to respond to the remaining input vectors of that particular input sequence. Thus, a different map node is allocated for every vector in the input sequence. Furthermore, as new vectors of the sequence come in, the activation of the previous winners decrease in proportion to a decay parameter *d* in the interval [0, 1]. The generalized training procedure of the SARDNET is presented in Table 1 below.

```
INITIALIZATION:
Clear all map nodes and activations to zero
TRAINING STEP:
LOOP: While not end of sequence
          Find inactivated BMU
          Assign 1.0 activation to that unit
          Adjust weight vectors of the nodes in the neighborhood
          Exclude the BMU form subsequent competition
          Decrement activation values for all other active nodes
END LOOP
Reset all activations
```

Table 1 The SARDNET training algorithm.

To recap, as in normal SOM weight vectors adapt to and become generalization of the input vectors $\boldsymbol{V_i}$. However, in the SARDNET each sequence $\boldsymbol{S}$ of length *l* is presented by *l* active nodes on the map, with their activity indicating the order in which they were activated. Thus, the map representation expands those areas of the input space that are visited most often during an input sequence.

2.2. Example: Mapping three-syllable words

The aim of the task was to map a set of three syllable words into SARDNET presentation. Each word consisted of sequence of phonemes, which were represented by five values. Furthermore, the length of the sequences varied from five to twelve phonemes. Totally 43 phonemes and three data sets consisting of 713, 988 and 1628 words were used in the experiment.

Maps of sizes from 16 to 81 units were trained with the all three data sets. With the chosen parameters, i.e., a learning rate α=0.45 and neighborhood radius decreasing from 5-1 to 0, the organization was quite fast. That is, each training session took only about 10 epochs. It turned out that the accuracy - the percentage of unique representations out of all word sequences - for all sets was better than 97.7%, which implies that the network manages to pack the input sequences very efficiently. In effect, the network is sometimes able to "reuse" map units to represent multiple similar phonemes resulting in the most descriptive representation of the data given the available resources. Furthermore, the topological organization of the map allows finding a good set of reusable vectors that can stand for different phonemes in different sequences making the representation more efficient.

Figure 2 displays representations of six different words. It can be seen that in general similar sequences have similar representations. Furthermore, since similar phonemes are mapped next to each other, the network is indeed topologically ordered and thus small changes in the input results in small changes in the representation. Also, it can be verified that the map units of the small map are reused to represent several different phonemes in different contexts. Finally, it can be stated that the network results in robust, dense and descriptive representations which is useful in real word applications with large amounts of incomplete or noisy data.

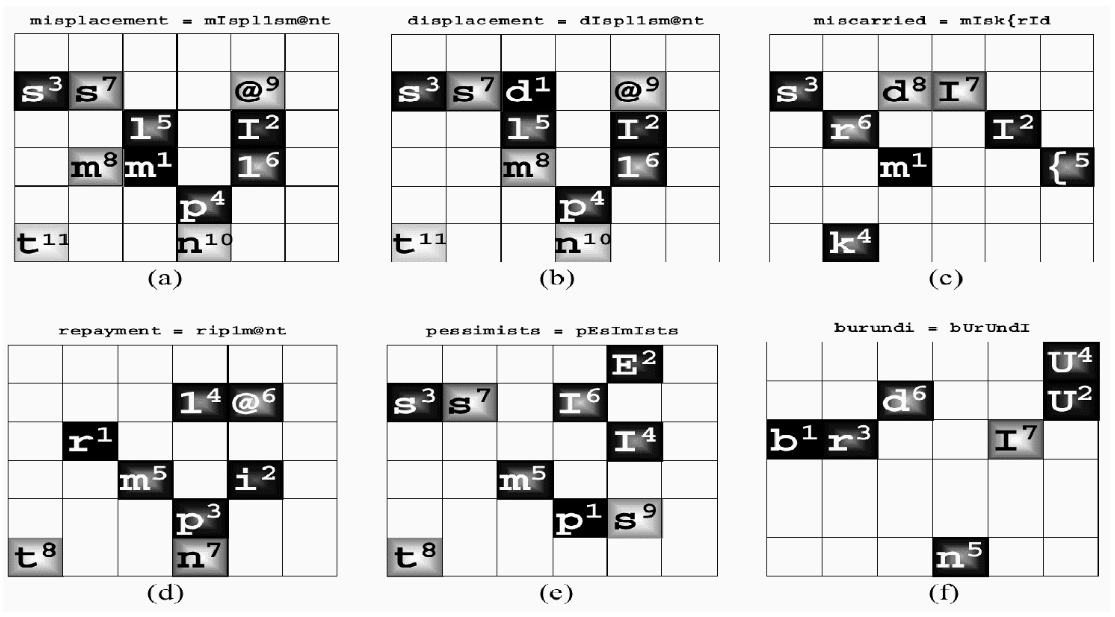

misplacement = mIspl1sm@nt
s^3 s^7 $@^9$ l^5 I^2 m^8 m^1 l^6 p^4 t^{11} n^{10}
(a)
displacement = dIspl1sm@nt
s^3 s^7 d^1 $@^9$ l^5 I^2 m^8 l^6 p^4 t^{11} n^{10}
(b)
miscarried = mIsk{rId
s^3 d^8 I^7 r^6 I^2 m^1 $\{^5$ k^4
(c)
repayment = rip1m@nt
l^4 $@^6$ r^1 m^5 i^2 p^3 t^8 n^7
(d)
pessimists = pEsImIsts
E^2 s^3 s^7 I^6 I^4 m^5 p^1 s^9 t^8
(e)
burundi = bUrUndI
U^4 d^6 U^2 b^1 r^3 I^7 n^5
(f)

Figure 2 The SARDNET representations of six words.

3. SARDSRN

SARDSRN is a hybrid feed-forward network designed for natural language processing tasks, especially shift-reduce parsing. Essentially it is a Simple Recurrent Network (SRN) augmented by the SARDNET. The main idea is to expand the limited sequence processing of SRN – e.g., poor handling of long-term dependencies of the input sequences – by explicitly representing the input sequence in the SARDNET. Thus the good generalization and cognitive properties of the SRN can be combined with exact and compressed input sequence representations of the SARDNET resulting in a robust system being able to handle input sequences with a complex structure.

3.1. Network architecture

The structure of the SARDSRN network is presented in Figure 3. The input data of the network are similar to that of the SARDNET. However, here also the desired output is available, so the network actually performs a supervised learning task. Regarding the structure, , the basic element of the network is the SRN, which consists of a present and previous hidden layers and an output layer. The SARDNET is only a part of network connecting to the hidden layer of the SRN. The operation of the network is such that the present hidden layer gets the input vector, the SARDNET representation of the input sequence thus far and the previous hidden layer. The output maps the information of the hidden layer into the desired output determined by learning, which is done through error back-propagation. The SARNET component of the network is trained in the unsupervised manner described in section 2.1. The overall training procedure of the SARDSRN is presented in Table 2.

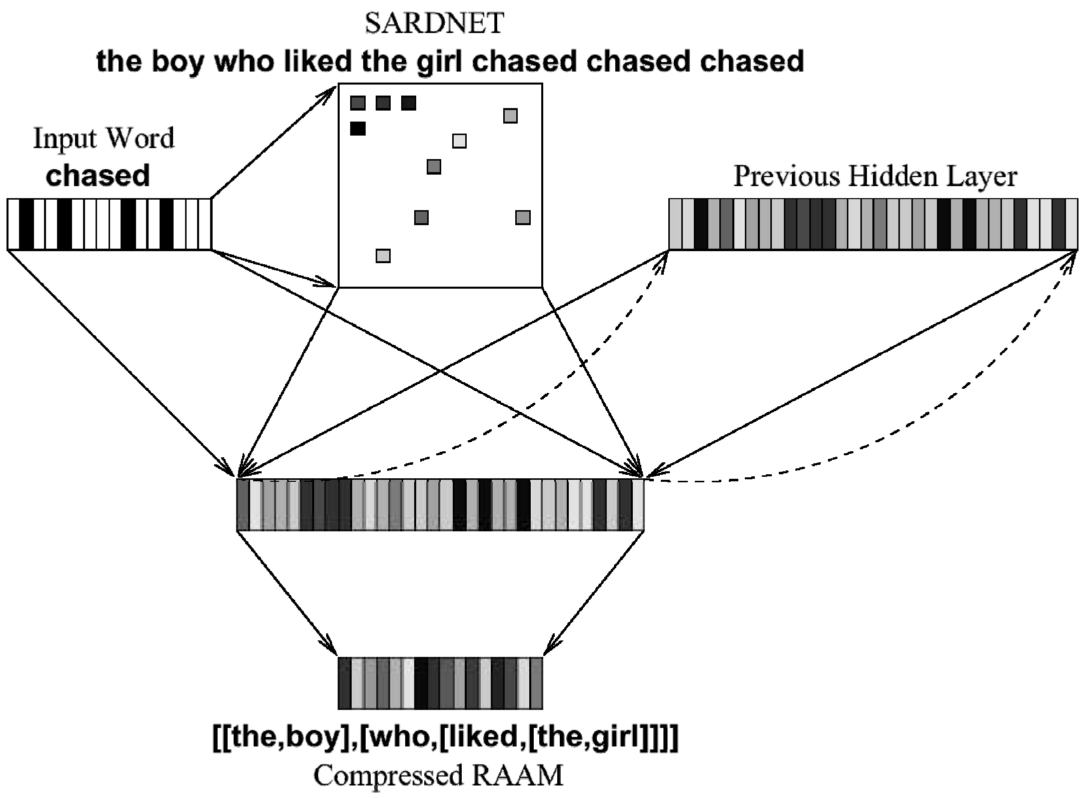

SARDNET
the boy who liked the girl chased chased chased
Input Word
chased
Previous Hidden Layer
[[the,boy],[who,[liked,[the,girl]]]]
Compressed RAAM

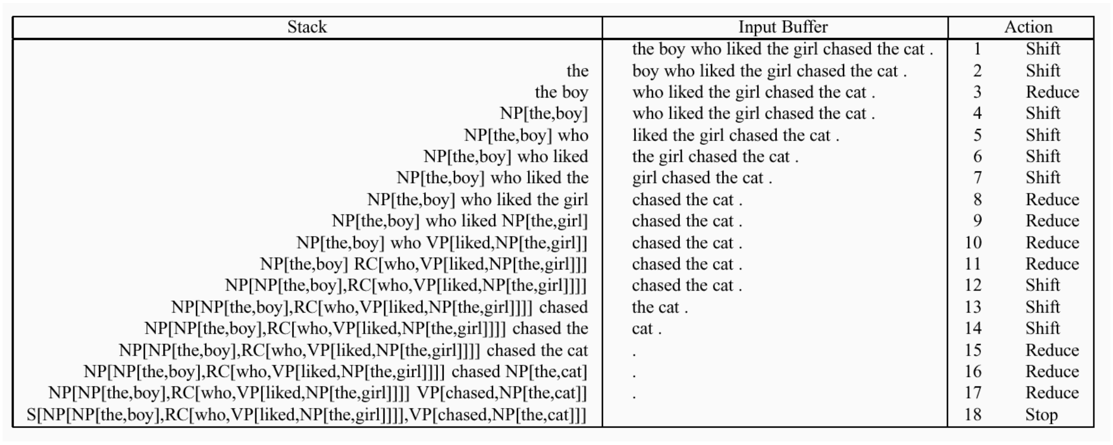
Figure 3 The SARDSRN architecture.

```
INITIALIZATION:
Initialize the SRN, e.g., with random values and the SARDNET as described in Table 1.
LEARNING STEP:
LOOP: While not end of sequence
        Do the SARDNET training.
        Feed the current input, the SARDNET representation of the input sequence thus far, and the previous hidden layer of the SRN into the present SRN hidden layer.
        Map the hidden to the output layer through error back-propagation training.
        Save the present hidden layer into the previous hidden layer.
END LOOP
Reset all activations of the SARDNET
```

Table 2 The SARDSRN training algorithm.

3.2. *Example: Shift-reduce parsing*

The purpose of the example was to parse sentences like presented in Figure 3. The input consisted of sequences of words, which were represented by 64-dimensional vectors. The sentences as well as the training targets corresponding to each step of the parsing process were generated by a phase structure grammar. The output layer of the network was actually a stack, which was implemented as a Recursive Auto-Associative Memory (RAAM). The network functioned so, that at each step either a word was shifted to the stack or one or more top elements of the stack were reduced into a new element (hence the name shift-reduce parsing). An example of parsing a sentence is illustrated in Figure 4, where each line represents a parser operation.



Stack	Input Buffer	Action	
	the boy who liked the girl chased the cat .	1	Shift
the	boy who liked the girl chased the cat .	2	Shift
the boy	who liked the girl chased the cat .	3	Reduce
NP[the,boy]	who liked the girl chased the cat .	4	Shift
NP[the,boy] who	liked the girl chased the cat .	5	Shift
NP[the,boy] who liked	the girl chased the cat .	6	Shift
NP[the,boy] who liked the	girl chased the cat .	7	Shift
NP[the,boy] who liked the girl	chased the cat .	8	Reduce
NP[the,boy] who liked NP[the,girl]	chased the cat .	9	Reduce
NP[the,boy] who VP[liked,NP[the,girl]]	chased the cat .	10	Reduce
NP[the,boy] RC[who,VP[liked,NP[the,girl]]]	chased the cat .	11	Reduce
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]]	chased the cat .	12	Shift
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]] chased	the cat .	13	Shift
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]] chased the	cat .	14	Shift
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]] chased the cat	.	15	Reduce
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]] chased NP[the,cat]	.	16	Reduce
NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]] VP[chased,NP[the,cat]]	.	17	Reduce
S[NP[NP[the,boy],RC[who,VP[liked,NP[the,girl]]]],VP[chased,NP[the,cat]]]		18	Stop

Figure 4 Shift-Reduce Parsing a sentence.

The total number of sentences generated was 436 and four data sets with 20%, 40%, 60% and 80% of the sentences was used for training. Each data set was trained four times. The training of the network was stopped when the validation done by a 22-sentence validation set began to level off. The size of the SARDNET used in the example was 144-units and the hidden and output layer had 200 and 64 units respectively. Other parameter values used were the learning rate of 0.2 for SRN and that of 0.5, the decay rate of 0.9, and the neighborhood decreasing from 6 to 0 for the SARDNET.

Regarding the results, the average mismatches, - a measure of the correctness of the information in the RAAM presentation - was used as a quality measure. So, on average there were only about three mismatches per sentence even in the most difficult case of having only 20% of the sentences as a training set. For the easiest case there were less than 0.5 mismatches. For comparison these results were checked against a standard SRN parser and it was found out that results of the SARDSRN were roughly an order of magnitude better that those of the SRN.

4. Discussion

The authors claim that with the SARDNET approach they can overcome the shortcomings of the previous approaches proposed to classification of sequential information. Furthermore, they stated that the dense representation together with the tendency to map similar sequences to similar representations should enable using the SARDNET in real-world applications. They also proved that in a small real-world problem. Finally they suggested that the approach can easily be modified to be used in the auto-association tasks, e.g., in sequence completion. However, in literature there have been presented doubts, that the network might have problems, at least in some applications, with input decoding and inputs having repeating subsequences.

Regarding the SARDSRN the authors claim that their network could improves the memory degradation problem of the SRN. They also stated that instead of being capable of handling only simple sequences with shallow structure (as is the case with the SRN) the proposed approach results in a system useful in more realistic applications, i.e., having longer and more complex sequences. They also showed that to be true in the Shift-Reduce Parsing case, at least to some extent. However, some other researchers have suggested that in practice, problems may occur with the considerable computational cost of the SARDNET. Furthermore, it have been claimed that the results of the network may be unreliable in the hard real world problems.

The authors have also extended the system further by replacing the SRN network by a Nonlinear AutoRegressive model with eXogenous inputs (NARX), which can better handle long-term dependencies. With this approach they claimed to have even better results without having to select appropriate number of delays, which is the major problem with the ordinary NARX.

5. References

[1] James, D. L. and Miikkulainen, R. SARDNET: A self-organizing feature map for sequences. In Tesauro G., Touretzky D. S. and Leen T. K. (Eds), *Advances in Neural Processing Systems*, pages 577-584, MIT Press, Cambridge, 1995.

[2] Mayberry III, M. R. and Miikkulainen, R. SARDSRN: A neural network shift-reduce parser. *Proc. 16th Annual Int. Joint Conf. on Artificial Intelligence* (IJCAI-99), pages 820-825, Stockholm, Sweden, 1999.

[3] Mayberry III, M. R. and Miikkulainen, R. Using a sequential SOM to parse long-term dependencies. *Proc. 21st Annual Conf. of the Cognitive Science Society* (COGSCI-99), pages 367-372, Vancouver, Canada, 1999.