Experimental assessment of connectionist regular inference from positive and negative examples

René Alquézar[(1)], Alberto Sanfeliu[(2)] and Miguel Sainz[(2)]
[(1)]Dep. de Llenguatges i Sistemes Informàtics, Univ. Politècnica de Catalunya (UPC)
Jordi Girona Salgado 1-3, Mòdul C5, 08034 Barcelona, Spain
[(2)]Institut de Robòtica i Informàtica Industrial, UPC - CSIC, Barcelona
alquezar@lsi.upc.es, asanfeliu@iri.upc.es, msainz@iri.upc.es

Abstract

In this paper, the ability of recurrent neural networks (RNNs) for regular inference (RI) from positive and negative examples is investigated. As in some previous works [1, 2], RNNs were trained to learn the string classification task from samples of some target regular languages. In addition, an automaton extraction method [3] was applied to each one of the trained nets to obtain a description of the inference outcome. For comparison purposes, a symbolic RI method, the RPNI algorithm [4], was also run on the same data. Although the automaton extraction step improved the generalization performance of the nets, the inference quality using RNNs was not so good as the one shown by the RPNI algorithm.

Key Words : grammatical inference, recurrent neural networks, machine learning.

1 Introduction

Regular inference (RI) is the problem of learning a regular language, or a finite-state automaton (FSA) accepting it, from string examples or queries. For the case of RI from positive and negative samples, a theoretical framework has been presented in [5], and several methods have been proposed, using symbolic [4, 6], connectionist [1, 2, 3, 7] and genetic [8] approaches. However, the comparative performance of these methods on benchmark tests has not been assessed sufficiently. In this work, both first- and second-order *Augmented Single-Layer Recurrent Neural Networks* (ASLRNNs)[1] [3, 7] have been trained to learn the string classification task from sparse samples containing positive and negative examples, and *unbiased FSAs* (or UFSAs)[2] have been extracted from the trained nets using an extension of a clustering method described in [3]. Likewise, the RPNI algorithm [4] have been applied to the same data for comparison purposes. The generalization performance of the trained ASLRNNs, the UFSAs extracted from them, and the FSAs inferred by the RPNI algorithm has been evaluated on common test samples. The fifteen regular languages used as benchmark in [8], which include the seven Tomita's languages tested in [1, 2], have been chosen for this study (see Fig. 1), together with the same samples that were employed as training and test sets, respectively, in [8].

2 Architectures and algorithms used in the tests

The results of some previous experiments on learning the next-symbol prediction task for RI from positive examples [9] were taken into account to select the types of RNNs for this study. Thus, first-order ASLRNNs including a fully-connected recurrent layer of N hidden units with the antisym-log activation function

[1] An ASLRNN consists of a fully connected recurrent layer (an SLRNN) augmented with a feedforward layer of output units, which receive the activations of the recurrent (hidden) units.

[2] An UFSA is a type of automaton with positive and negative final states, which has been proposed in [3, 6] to represent positive and negative information in a symmetrical way.

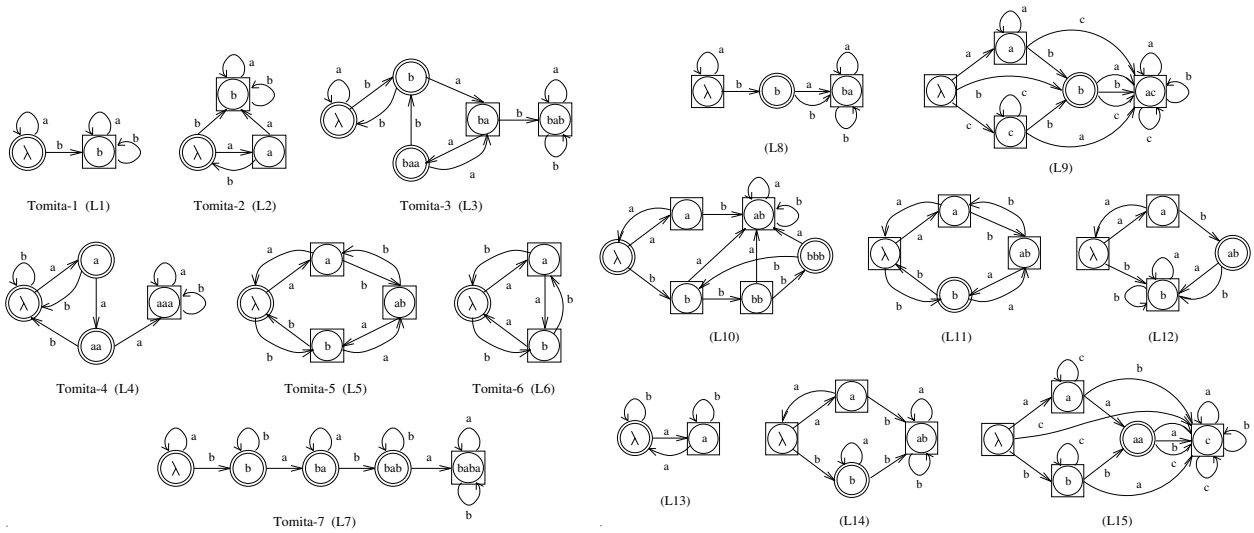

Figure 1. *Minimal-size DUFAs (deterministic UFSAs) for the* 15 *test languages.*

$$g_1(\sigma) \;=\; \mathrm{sgn}(\sigma)\;\log(1+a|\sigma|) \tag{1}$$

and an output layer of just one unit with the sigmoid activation function

$$g_2(\sigma) \;=\; \frac{1}{1+e^{-a\sigma}}, \tag{2}$$

were preferred with respect to first-order SLRNNs or other ASLRNNs with sigmoid units in the recurrent layer, since ASLRNNs with the (antisym-log, sigmoid) pair of functions had learned much better in [9]. It was also decided to test both first- and second-order ASLRNNs (with identical activation functions and number of units), because RNNs with second-order connections had been used successfully for the string classification task by other researchers (e.g. [2]). The constant a in Eqs. (1) and (2) was set respectively to 2 for g_1 and 1 for g_2.

The dynamic behavior of an ASLRNN is described by

$$y_k(t) \;=\; g_1\;\left(\sum_{i=1}^{M} w_{ki}\;x_i(t)\;+\;\sum_{j=1}^{N} w_{k(M+j)}\;y_j(t-1)\right) \quad \text{for}\;\; 1\le k\le N, \tag{3}$$

for a first-order recurrent layer, or

$$y_k(t) \;=\; g_1\;\left(\sum_{i=1}^{M}\sum_{j=1}^{N} w_{kij}\;x_i(t)\;y_j(t-1)\right) \quad \text{for}\;\; 1\le k\le N, \tag{4}$$

for a second-order recurrent layer, where $x_i,\; 1\le i\le M$, and $y_j,\; 1\le j\le N$, refer respectively to network inputs and hidden unit activation values; and by

$$O_l(t) \;=\; g_2\;\left(w^o_{l0}\;+\;\sum_{j=1}^{N} w^o_{lj}\;y_j(t)\right) \quad \text{for}\;\; 1\le l\le P, \tag{5}$$

for the output layer, where $O_l,\; 1\le l\le P$, refer to the activation values of the output units.

A single output unit is enough for the string classification task [2, 7] (see Fig. 2), where the net acts like an acceptor depending on the output unit value at the end of a string presentation

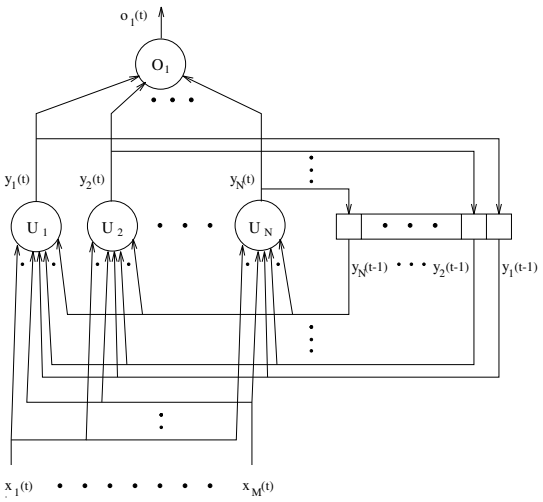

Figure 2. *A generic ASLRNN architecture for regular inference through classification of positive and negative examples.*

$O_1(t_f)$. Thus, a set of string-response pairs are given to the net at each training epoch, where the response corresponds to the desired value of $O_1(t_f)$, e.g. $T_{accept} = 1$ for positive strings and $T_{reject} = 0$ for negative strings. A string is correctly classified by the net if the absolute difference between the desired response and $O_1(t_f)$ is less than a tolerance threshold ϵ.

A true gradient-descent learning algorithm was used to train the ASLRNNs, such that back-propagation was applied to update the weights of the output unit and supply the error values for the hidden units, and either (first-order) Schmidhuber's version of RTRL algorithm [10] (with an average time complexity per time step of $O(N^3)$) or an adaptation of it to second-order nets (of $O(N^3M)$) was applied to train the recurrent layer. The learning rate α and the momentum parameter β were fixed for all the runs: $\alpha = 0.025$ and $\beta = 0.5$.

In order to extract a consistent *deterministic UFSA* (DUFA) from an ASLRNN previously trained to classify a sample $S = (S^+, S^-)$, an extension of the algorithm described in [3] was used. A hierarchical clustering on the space of hidden unit activations serves to guide a state merging process from the prefix tree of the whole sample S, $PTU(S)$. At each step, a maximum of k pairs of clusters (states), which are the closest in the activation space, are selected as candidates to be merged. The deterministic merge of the states associated with the first pair in the ordered list that leads to a consistent DUFA is carried out. The clustering procedure ends when all candidate pairs lead to inconsistent DUFAs. The DUFA resulting of this process is minimized to obtain the extracted DUFA. The parameter k was set to 50 for the tests; in this way, it was expected that a maximal generalization of $PTU(S)$ would be obtained in most cases[3].

3 Benchmark data, test procedure and results

Fig. 1 shows the target minimal-size DUFA for each one of the 15 test languages [8]. Ten different samples of each language were used as training sets. Each sample $S = (S^+, S^-)$ was structurally complete with respect to the corresponding minimal-size DUFA. The original samples generated by Dupont [8], which contained repeated strings, were slightly modified just to include the empty string λ in S^+ or S^- appropriately[4]. The average number of different examples in the samples ranged from 16 for L_1 to 71 for L_{10}. The number of recurrent hidden units that were included in

[3] For $k = 50$, the extracted DUFA is guaranteed to meet this property if the final number of clusters is ≤ 10.

[4] In addition, ten negative samples were generated for language $L_1 = a^*$, since Dupont considered L_1 to be defined over a unary alphabet, and therefore, he did not include any negative sample for it [8].

both the first- and second-order ASLRNNs was set to $N = 4$ for languages $L_3, L_7, L_9, L_{10}, L_{15}$ (for which the minimal DUFA contains 5 or 6 states), and $N = 3$ for the rest of test languages (for which the minimal DUFA contains up to 4 states)[5]. Two trials, with different random initial weights in the interval [-0.25,0.25], were run for each training set and each network type, so that a total number of 20 runs were performed with each model for each test language.

Each network was trained to classify the strings in the given training set, which were presented by alternating positive and negative examples until exhaustion of one of the sets and then giving the rest of examples. A local encoding was used to represent the language symbols in the input signals. The recurrent-unit activations were reset to 0.1 at the start of each string presentation, and no end-of-string symbol was used. The training phase ended when all the strings in the training set were correctly classified, using $\epsilon = 0.1$ as tolerance threshold, and a minimum number of $1,000$ epochs were performed[6]. Then, a consistent DUFA was extracted from the final net dynamics on the given sample by applying the clustering algorithm aforementioned.

To get an idea of the running time required, $2,500$ training epochs over one of the largest samples (L_{15} sample #10), with 79 strings and a total length of 274 symbols, took 1 minute of CPU time on a DEC Alpha Station 3400 computer for a first-order ASLRNN and 2 minutes 10 seconds for a second-order ASLRNN, both nets with $N = 4$ hidden units and 1 output unit. The corresponding DUFA extraction, starting on a prefix tree UFSA with 85 states and ending on a DUFA with 15 states, took around 3 seconds of CPU time.

The same test samples that had been used in [8] were employed. This is, for each run, all the strings up to length l but the given training examples were included, where l was set to 9 or 7 for target languages over $\{a, b\}$ or $\{a, b, c\}$, respectively. After training, the test strings were classified both by the final net and the extracted DUFA. A test string was accepted by a trained net if $O_1(t_f)$ was greater than 0.5, and it was rejected otherwise (i.e. $\epsilon = 0.5$ was used in this step). For both ASLRNNs and extracted DUFAs, the correct classification rates on the three sets of positive, negative, and all test strings were computed.

For each language and each type of inference, the averages of the above rates over the 20 runs were calculated, and they are displayed in the former three wide columns of Tables 1 and 2. The fourth wide column refers to the arithmetic mean of the positive and negative classification rates. The fifth wide column in Table 1 shows the percentage of times the whole test sample was correctly classified by the trained net (success rate); whereas in Table 2, it shows the percentage of times the target DUFA was extracted (identification rate). The bottom row of the tables displays the averages of these five features over the 15 test languages.

The results of the generalization tests are presented in Table 1, for the trained first- and second-order ASLRNNs, and in Table 2, for the corresponding extracted DUFAs. It can be observed that both architectures performed rather similarly, a little bit better the second-order ASLRNNs. However, all global classification rates were below the 80%, and the success rate was very poor (7%), since for most of the languages, a perfect classification of the test sample was not achieved in any of the 20 runs. In general, the DUFAs performed better than the trained nets from which they were extracted, improving the global classification rates between a 3% and a 4%, and raising the identification rate up to a 19%.

Table 3 shows the summary of the global results obtained by the different RI methods tested in the experiment. The best performance corresponded to the RPNI algorithm. Dupont reported average correct classification rates of 85.4% and 94.4% for his non-incremental and semi-incremental genetic RI methods, respectively, on this benchmark [8].

[5]The results of the experiments reported in [2] were used to set the value of N. The aim was to provide sufficient hidden units to allow learning of training samples while not impairing the generalization performance.

[6]The requirement of a minimum number of training epochs was aimed at giving the net enough time to contract the clusters formed in the hidden unit activation space.

Nets	Pos.class		Neg.class		Tot.class		Av.class		Success	
	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.
L_1	90.0	86.0	99.6	96.5	99.5	96.4	94.8	91.3	45.0	35.0
L_2	92.1	84.5	92.6	89.7	92.6	89.7	92.4	87.1	20.0	10.0
L_3	53.2	78.5	79.3	72.9	68.8	74.9	66.3	75.7	0.0	0.0
L_4	66.6	74.7	67.6	58.8	67.0	68.0	67.1	66.8	0.0	0.0
L_5	49.0	52.4	74.1	76.6	69.9	72.5	61.6	64.5	0.0	5.0
L_6	37.1	44.0	63.8	60.4	54.9	54.9	50.5	52.2	0.0	0.0
L_7	59.2	77.4	60.7	45.5	60.1	57.0	60.0	61.5	0.0	0.0
L_8	82.3	84.6	89.6	72.0	89.5	72.2	86.0	78.3	10.0	0.0
L_9	87.7	90.0	92.9	81.1	92.9	81.2	90.3	85.6	0.0	0.0
L_{10}	46.0	53.7	93.7	93.4	92.7	92.6	69.9	73.6	0.0	0.0
L_{11}	69.2	76.0	55.8	70.2	60.3	72.1	62.5	73.1	0.0	5.0
L_{12}	94.3	88.3	96.4	90.7	96.3	90.6	95.4	89.5	0.0	5.0
L_{13}	50.7	81.1	51.1	81.1	50.9	81.1	50.9	81.1	0.0	45.0
L_{14}	39.1	65.3	85.2	79.7	83.7	79.2	62.2	72.5	0.0	0.0
L_{15}	58.0	75.9	89.7	84.5	89.5	84.4	73.9	80.2	0.0	0.0
Mean	65.0	74.2	79.5	76.9	77.9	77.8	72.3	75.5	5.0	7.0

Table 1. *Classification results of first-order (left) and second-order (right) ASLRNNs.*

DUFAs	Pos.class		Neg.class		Tot.class		Av.class		Identif.	
	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.	F.O.	S.O.
L_1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
L_2	96.7	93.7	96.4	94.4	96.4	94.4	96.6	94.1	55.0	25.0
L_3	47.3	59.4	89.5	85.3	72.8	75.0	68.4	72.4	0.0	5.0
L_4	55.6	53.8	84.5	82.4	67.6	65.7	70.1	68.1	5.0	10.0
L_5	38.3	42.4	82.4	82.8	75.0	76.1	60.4	62.6	0.0	15.0
L_6	31.0	31.6	70.8	76.1	57.5	61.2	50.9	53.9	0.0	0.0
L_7	53.8	61.4	71.3	71.0	64.9	67.4	62.6	66.2	0.0	10.0
L_8	100.0	90.0	96.4	81.1	96.5	81.2	98.2	85.6	70.0	10.0
L_9	96.0	91.7	95.6	88.8	95.6	88.8	95.8	90.3	5.0	0.0
L_{10}	54.2	55.6	93.5	94.7	92.7	93.9	73.9	75.2	0.0	0.0
L_{11}	59.8	65.3	62.9	75.8	61.8	72.3	61.4	70.6	0.0	20.0
L_{12}	100.0	96.5	97.6	93.9	97.6	93.9	98.8	95.2	40.0	30.0
L_{13}	41.3	78.7	71.4	87.0	56.3	82.9	56.4	82.9	10.0	65.0
L_{14}	39.3	61.8	91.1	83.7	89.5	83.0	65.2	72.8	0.0	0.0
L_{15}	72.1	75.3	93.2	93.4	93.0	93.2	82.7	84.4	0.0	0.0
Mean	65.7	70.5	86.4	86.0	81.1	81.9	76.1	78.3	19.0	19.3

Table 2. *Classification results of DUFAs extracted from trained ASLRNNs.*

4 Conclusions

First- and second-order ASLRNNs have demonstrated similar generalization capabilities on the benchmark test, somewhat better the latter, with approximately a 78% of total correct classification rate and full success on the test sample only in a 7% of the runs. These results were improved by the extracted DUFAs, which reached an 82% of total correct classification rate on the test samples and a 19% of full success (and target identification) rate. This validates empirically the UFSA extraction method proposed and confirms the beneficial effects, pointed out by other researchers [2], of extracting an automaton from a trained RNN with continuous activation function. Not only a symbolic representation of the inference result is obtained, but also the generalization performance is improved and the problem of bad classification of long strings due to drifting activations is avoided. However, even after DUFA extraction, the inference quality shown by the connectionist RI methods tested has been notably worse than that displayed by the symbolic RPNI method [4] (96% of total correct classification rate and 78% of identification rate). The average correct classification rates of the connectionist approaches have been also worse than the ones reported by Dupont for his genetic RI approaches [8].

RGI METHOD	Pos.class	Neg.class	Tot.class	Av.class	Success
F.O. ASLRNNs (An.log, Sigm.)	65.0	79.5	77.9	72.3	5.0
S.O. ASLRNNs (An.log, Sigm.)	74.2	76.9	77.8	75.5	7.0
DUFAs F.O.nets (An.log, Sigm.)	65.7	86.4	81.1	76.1	19.0
DUFAs S.O.nets (An.log, Sigm.)	70.5	86.0	81.9	78.3	19.3
RPNI (Oncina-García)	91.8	97.5	96.0	94.7	78.0

Table 3. *Summary of results of the regular inference experiments.*

Nevertheless, it might be argued that the benchmark samples were specially suitable to the RPNI method, since they often included the representative samples that allowed the identification of the target automata by this algorithm. On the other hand, the small number of examples in the benchmark training sets (an average of 37 different strings) might be insufficient, from the statistical point of view, to allow the networks generalize correctly, this being probably the cause of the middling results obtained by the RNN-based approaches. Moreover, we conjecture that a quite higher generalization performance might be achieved by ASLRNNs by increasing enough the size of the training set. This hypothesis is supported by the results reported in some previous studies on inferring the Tomita's languages using RNNs; e.g., an average of 56% total correct classification rate on test samples after learning to classify small training sets of around 20 examples [1], but a 99.9% when training sets of around 1,000 examples were used [2]. Hence, it seems that, at least for the tested task and languages, connectionist RI methods need larger samples than symbolic RI methods to reach a similar level of generalization performance. This is not surprising, since connectionist methods can be regarded indeed as statistical approaches.

References

[1] R.L. Watrous and G.M. Kuhn, "Induction of finite state languages using second-order recurrent networks," *Neural Computation* **4**, pp.406-414, 1992.

[2] C.B. Miller and C.L. Giles, "Experimental comparison of the effect of order in recurrent neural networks," *Int. Journal of Pattern Recognition and Artificial Intelligence* **7** (4), pp.849-872, 1993.

[3] R. Alquézar and A. Sanfeliu, "A hybrid connectionist-symbolic approach to regular grammatical inference based on neural learning and hierarchical clustering," in *Grammatical Inference and Applications*, R.C.Carrasco and J.Oncina (eds.), Springer-Verlag, LNAI 862, pp.203-211, 1994.

[4] J. Oncina and P. García, "Identifying regular languages in polynomial time," in *Advances in Structural and Syntactic Pattern Recognition*, H.Bunke (ed.), World-Scientific, Singapore, pp.99-108, 1992.

[5] P. Dupont, L. Miclet and E. Vidal, "What is the search space of the regular inference?," in *Grammatical Inference and Applications*, R.C.Carrasco and J.Oncina (eds.), Springer-Verlag, LNAI 862, pp.25-37, 1994.

[6] R. Alquézar and A. Sanfeliu, "Incremental grammatical inference from positive and negative data using unbiased finite state automata," in *Shape, Structure and Pattern Recogniton*, D.Dori and A.Bruckstein (eds.), World Scientific Pub., Singapore, pp.291-300, 1995.

[7] A. Sanfeliu and R. Alquézar, "Active grammatical inference: a new learning methodology," in *Shape, Structure and Pattern Recogniton*, D.Dori and A.Bruckstein (eds.), World Scientific Pub., Singapore, pp.191-200, 1995.

[8] P. Dupont, "Regular grammatical inference from positive and negative samples by genetic search: the GIG method," in *Grammatical Inference and Applications*, R.C.Carrasco and J.Oncina (eds.), Springer-Verlag, LNAI 862, pp.236-245, 1994.

[9] R. Alquézar and A. Sanfeliu, "Inference and recognition of regular grammars by training recurrent neural networks to learn the next-symbol prediction task," in *Advances in Pattern Recognition and Applications*, F.Casacuberta and A.Sanfeliu (eds.), World Scientific Pub., Singapore, pp.48-59, 1994.

[10] J. Schmidhuber, "A fixed size storage O(n^3) time complexity learning algorithm for fully recurrent continually running networks", *Neural Computation* **4**, pp.243-248, 1992.