

Rethinking Rule Extraction from Recurrent Neural Networks

Henrik Jacobsson and Tom Ziemke

University of Skövde, School of Humanities and Informatics

P.O. Box 408, SE-541 28 Skövde, Sweden

{henrik.jacobsson,tom.ziemke}@his.se

Abstract

We will in this paper identify some of the central problems of current techniques for rule extraction from recurrent neural networks (RNN-RE). Then we will raise the expectations of future RNN-RE techniques considerably and through this, hopefully guide the research towards a common goal. Some preliminary results based on work in line with these goals, will also be presented.

1 Introduction

The problem of extracting rules, or finite state machines from recurrent neural networks (RNN-RE) has occupied several researchers on and off during the last 15 years. The achievements of this research has recently been compiled into a review on the subject [Jacobsson, 2005]. Unlike the field of neural networks in general, the field of RNN-RE has not actually developed much since it was first conceived. The first algorithm published [Giles *et al.*, 1992] is also still the most cited and seems to be the most commonly used algorithm although it is quite simple in its nature. More recent methods seem not to have been built on the findings of earlier approaches and there are no common benchmarks for how to compare these algorithms, nor are there any attempts to compare them in the first place. We would hold that this is due to the lack of a common goal in this field.

In this paper, we will first present some thoughts on why we at all can expect to be able to extract rules from an RNN in the first place, and why these properties should force us to rethink how connectionists should conduct and present their research. Some common constituents of existing RNN-RE techniques will be discussed briefly together with a analysis of what is lacking in these algorithms. Some preliminary results that have been obtained by an attempt to solve these problems will also be presented briefly. Then we will try to extrapolate from the seeds of existing solutions, a set of goals that may guide research efforts to grow beyond the original intention of just extracting rules from networks.

2 The golden properties of RNNs

If we compare the study of RNNs with the study of physical dynamic systems, there are some quite obvious differences

that, comparably, make RNNs perfect subjects for systematic analysis. Let us call these the “golden properties” of RNNs (properties that certainly are shared by a broad range of other systems too). Since RNNs are computer simulated systems, they allow us to (among other things):

- reproduce results with arbitrarily high accuracy,
- create more networks of the same kind without much additional effort after the framework for the creation of the first network is implemented, giving us an easy access to the population of all possible networks,
- duplicate networks (and distribute them among research colleagues),
- study the effect of damage to the network under controlled conditions,
- do nonperturbative studies of internal properties to an arbitrary degree of detail.

In other words, RNNs are almost perfect experimental subjects. Very few scientific communities have the luxury of studying entities with properties so inviting for conducting research on them.

These would be golden properties of a phenomenon for any scientist. And this puts us connectionists, who are scientifically investigating these networks on a ledge: If we are studying a phenomenon with these inviting properties, do we not also have an obligation to utilise these properties as best we can? We would say that we do. Connectionists should not be content with the limited precision associated with the limited resources associated with research on physical systems. Let us exemplify this: If an astronomer wants to verify a theory on, for example, the frequency of earth-like planets in the universe, he or she must rely on secondary or tertiary data from various sources. Gathering new data is expensive and tedious, and astronomers treat their rare data with great care because of this. Often astronomical theories have large variation because of the lack of high-quality and non-contradicting data, e.g. that the estimated age of the universe is between 10 and 20 billion years. A neural network researcher, on the other hand, should not be satisfied with secondary data, since the networks themselves can be duplicated from the original source. In case of quantitative theories, he or she should also not be satisfied to verify these theories on only a few instances

of networks since there is no limit to the number of networks that can be generated.

The biggest problem is of course that the RNN researcher then “drowns” in all the data from the networks. How can one get an overview over networks when they are all individual entities with potentially quite complex behaviours?

And this is where rule extraction comes in (it comes into play under other circumstances as well, but let us stick to the analysis problem for now). Through rule extraction, the researcher can let some parts of the analysis of the individual networks be automated. If successful, the extracted deterministic rules can be enumerated, enlisted and further analysed in ways virtually impossible to do on the networks directly. And it is the golden properties that are precisely what allows us to do rule extraction from artificial neural networks at all. To do it on physical dynamic systems, e.g. a biological neuron or a star system, is by far much more difficult since it does not have these properties. But, for RNNs, we can build algorithms that utilises these properties and automate parts of the analysis process for the connectionist.

We are not there yet, however. Existing RNN-RE algorithms are still not reliable or efficient enough to be put into use in this manner. Let us have a quick look at why this is so.

3 Previous approaches and their deficiencies

What is an RNN-RE algorithm composed of? In [Jacobsson, 2005], four common ingredients were identified:

1. quantisation of the continuous state space of the RNN, resulting in a discrete set of states,
2. state and output generation (and observation) by feeding the RNN input patterns,
3. rule construction based on the observed state transitions,
4. rule set minimisation.

These four constituents are often quite distinguishable in the algorithms. For example, in [Giles *et al.*, 1992] (1) a simple grid partitioning of the state space quantised it, (2) states were generated by a breadth first search, (3) the rules were constructed by transforming the transitions in the quantised space into a deterministic finite automata and (4) the rules were minimised using a standard minimisation algorithm. Another example is [Tiño and Vojtek, 1998] where (1) a self organising map was used to quantise, (2) states were generated by observing the network in interaction with its domain and (3) stochastic rules were induced from these observations (no minimisation in this case).

In the corpus we can find up to eight different quantisation algorithms. There are no studies that compare any of these quantisers in the domain of RNN-RE. This means, in effect, that we still do not know which of the existing quantisers should be preferred. But the lack of experimental comparisons is actually not the main problem of these approaches.

The main problem is that none of the eight tested quantisation functions have been tailor-made to comply with the specific demands of quantising the state space of a dynamic system, where the state is recursively enfolded onto itself in interaction with a domain. The eight quantisers all build (roughly) on the assumption that spatial neighbours should

be merged and spatial strangers separated. But, two states that are very similar, spatially in the state space, may be very different from each other, functionally [Sharkey and Jackson, 1995].

4 Some preliminary results

When studying the earlier approaches we came to the conclusion that their main problem is the lack of integration of the four ingredients, state generation and rule construction and minimisation. Specifically the quantiser should take into account the dynamics of the RNN through closer integration with the other constituents, so that the state space is quantised based on its functional context as set of a states of a dynamic system in interaction with a domain.

This insight was the ground for the development of a novel algorithm named `CrySSMEx` (Crystallizing Substochastic Sequential Machine Extractor) which builds on a novel quantisation algorithm (a Crystalline Vector Quantiser, CVQ) which can merge and split state quanta based on their dynamical properties in the RNN. The main outline of the algorithm is described in Figure 1. By the introduction of `CrySSMEx`, a novel form of state machine, a substochastic sequential machine (SSM) is also introduced. SSMs can, unlike earlier rules, take into account that some information may be missing in the data collected from the RNN in interaction with its domain. `CrySSMEx` is parameter free and gradually eliminates indeterminism in the generated SSMs. Unfortunately, the constituents of the algorithm are quite complex (especially the test of equivalence of states in the SSM and the selection of data for performing splits) and there is no room for these details here¹.

`CrySSMEx` has been successfully applied on various networks in different domains. [Casey, 1996] showed that it is in principle always possible to extract finite state machines from RNNs that can recognise regular languages and this is verified in that using `CrySSMEx` on RNNs adapted to regular language domains requires very few iterations in the algorithm, and poses no real challenge. It is also quite easy to extract from successful networks predicting sequences of augmented strings, in random order, from the truncated context free language $0^n 1^n$ with $n \leq 10$ in about five to ten iterations. `CrySSMEx` has also been tested on networks with small random weights, a type of network shown to in theory be approximable by definite memory machines [Hammer and Tiño, 2003]. Figure 2 shows a typical “what excites node X ”-machine, i.e. with output symbols depending on the delta value of a specific node’s activation, extracted from an RNN with one input node, 10^3 state nodes and one output node, activation function $1/(1 + \exp(-net))$ with weights uniformly distributed in $[-0.1, 0.1]$ (Ω based on 10^4 random input symbols). It is also possible to extract nondeterministic stochastic machines, to a precision limited by invested computational resources, from chaotic dynamic systems, e.g. random RNNs

¹An open source distribution and an article on `CrySSMEx` are in preparation at the time of submission of this paper. The reason we bring up `CrySSMEx` here is not to present the algorithm as such, but to present it as a promising first step towards the Empirical Machine suggested in the last section of this paper.

$\text{CrySSMEx}(\Omega, \Lambda_i, \Lambda_o)$

Input: Time series data from the RNN, Ω , an input quantisation function, Λ_i , and an output quantisation function, Λ_o .

Output: A deterministic machine M .

begin

Let M be the stochastic machine resulting from an unquantised state space (i.e. only one state);

repeat

Select data for splitting indeterministic states;
 Split quanta according to split data;
 Create M with the new state quantiser, Λ_i and Λ_o as basis for quantisation;
if M has equivalent states **then**
 | Merge equivalent states;
end

until M is deterministic;

return M ;

end

Figure 1: A simplified description of the main loop of CrySSMEx . The machine M is created from the observed data contained in Ω by quantisation of input, output and state space, of which the latter is optimised in the algorithm.

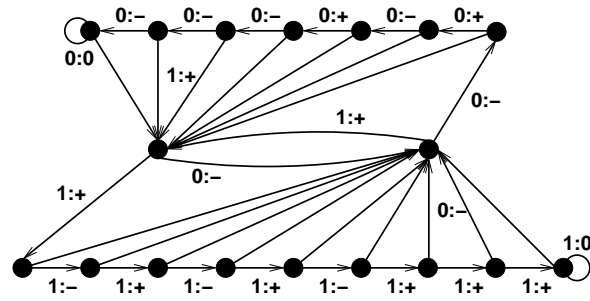


Figure 2: An extracted Mealy machine with two input symbols 0 and 1 and three output symbols $+$, $-$ and 0 representing that the value of the output node increases, decreases or remains the same (due to limited machine precision), respectively.

- have a clearly defined input, state and output, i.e. less or randomly structured RNNs may be problematic,
- have discrete input and output,
- have a fully observable state, otherwise unobserved state nodes or noise in the observation process would disturb the extraction process since the state space would not be reliably quantised,
- have state nodes that can be set explicitly (for search-based techniques),
- are deterministic, otherwise the same problem as if the state is not fully observable would occur,
- have certain dynamical characteristics, e.g. networks with chaotic behaviour are especially difficult to do rule extraction on,
- are fixed during RE, i.e. no training can be allowed during the RE process.

with larger weights, $0^n 1^n$ -RNNs tested on strings longer than the RNN can predict correctly, or other non-RNN dynamic systems.

5 Suggested goals and ambitions

What should the ambition of our development if future RNN-RE algorithms be? It may sound as a question with one obvious answer: “to generate rules that mimic and explain instantiations of RNNs”. But that answer builds on the assumption that it may not succeed, because if we thought that RNN-RE will actually succeed, to a satisfactory degree, the answer would be something like “to generalise RNN-RE algorithms so these algorithms will be applicable to as many problems as possible”. We firmly believe that CrySSMEx may be one possible step towards a fully functional RNN-RE technique that may satisfy the first modest ambition. Why then, if CrySSMEx or any other technique could be shown to work satisfactory as an RNN-RE, would we be satisfied with only extracting rules from RNNs, when there are a multitude of phenomena with similar functional structures as RNNs?

There may however be no yellow brick road towards such a goal. Apart from the problem discussed earlier, there are some implicit requirements RNN-RE algorithms have on the underlying RNN. [Craven and Shavlik, 1999] suggested that a rule extraction algorithm should not even be built on the assumption that it is analysing a network at all. But exactly how generic can these algorithms become? We probably need some of the golden properties of the RNNs as research objects and there are also some implicit requirements on the underlying system too since current RNN-RE techniques (including CrySSMEx) are preferably used on RNNs that:

- operate in discrete time, i.e. continuous-time RNN will not work,

Apart from the above requirements, computational complexity issues put limits on the number of state nodes and input patterns etc. These requirements are brought up here since the goals of future algorithms are naturally linked to the limitations of current algorithms in that one of the goal must be to eliminate them. If we intend to apply RNN-RE algorithms on a wider range of phenomena than just RNNs, the consequence is that our goals must be centred around creating algorithms that are as generic as possible.

Exactly what class of phenomena descendants of RNN-RE-algorithms can be used on remains an open issue, but at least we should be able to make a “wish-list” of what the algorithms should be able to do (we will still speak of RNNs, but feel free to substitute RNN by “dynamic system”, or, if you are optimistic, “physical system”):

- User freedom: The user should have the freedom to select, as parameters in one and the same algorithm, to prioritise between the following four, possibly opposite goals [Jacobsson, 2005]:
 - Comprehensibility, i.e. readability and rule set size.
 - Fidelity, i.e. the degree to which the rules actually mimic the underlying RNN.

- Accuracy, i.e. how well the rules generalise correctly on the domain of the RNN (this of course presupposes the existence of a domain in which “correctness” can be defined).
- Efficiency, i.e. how fast the rules are generated.
- Consistency over parameters: The results should not vary with parameters that are not of the kind described above. Preferably there are no other parameters than those that are of relevance for user preferences.
- Any-time extraction [Craven and Shavlik, 1999]: The RNN-RE algorithm should provide be able to provide a quick and dirty model which is then refined to a degree (towards one of the three first goals above) proportional to the computational power invested.
- Distance measure: The RNN-RE should provide means for testing similarity of RNNs. The similarity of two RNNs may not be connected to their weights or topology in any obvious way. A distance measurement also subsumes an equivalence tester, i.e. if $d(RNN_1, RNN_2) = 0$ then RNN_1 is equivalent with RNN_2 .
- Automatic subsystem identification: If a single RNN can be more efficiently described as several loosely connected subsystems, the RNN-RE should be able to discover this. This may be the case if the RNN uses separate subparts of itself to solve different subproblems in the domain.
- Queryable rules: Many times the comprehensibility of the rules overshadows all other ambitions with RNN-RE algorithms. But one can argue that a large, incomprehensible rule set, from which meaningful answers can be derived through queries, can be very useful. For example, if the network operates in a domain, it could be interesting to “ask the rules” to describe the circumstances under which the RNN makes mistakes. The answer to such queries should, for example, be helpful in designing the training conditions for the RNN.
- Automated empirical process: The RNN-RE should be able to discover when rules are not supported with enough data and then interact with the RNN such that the lacking data is acquired from the RNN. If this requirement is satisfied we propose to call the RNN-RE an *Empirical Machine* (not to be confused with the extracted machines themselves).
- Automated theory building: The RNN-RE should be able to create “theories” of one or more RNNs and strive for as universal and precise, and thereby falsifiable, theories [Popper, 1959], as possible. At this level we propose the RNN-RE should be called a *Popperian Machine*².

The above list is not sorted, but clearly the three last points build on each other. For the Empirical Machine to work, it has to be able to itself query the rules about how to interact with the RNN to collect missing data. The Popperian Machine may require a whole set of cooperating empirical machines generating, confirming and falsifying theories (which

are statements that must in turn be translated into queries) through “experiments” on the underlying systems.

The Empirical and Popperian machines are the ultimate goals we suggest for the field of RNN-RE. If we can achieve them it would mean a great deal for the research on RNNs in general, since many portions of the research can then be automated. If we at the same time can be able to apply descendants of RNN-RE algorithms on real physical systems, that would be a significant step towards an “artificial scientist”. The reason we see the Empirical and Popperian Machines as natural goals for future RNN-RE algorithms is that current techniques already contain fragments of these aspects; RNN-RE techniques, efficient or not, build detailed models (or theories) grounded in data, using a semi-empirical observational process. The preliminary promising results of CRYSSMEX, in which the empirical process has been made explicit through a model-based selection of observed data, suggests that this may be a reasonable way towards success.

References

- [Casey, 1996] M. Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178, 1996.
- [Craven and Shavlik, 1999] M. W. Craven and J. W. Shavlik. Rule extraction: Where do we go from here? Technical Report Machine Learning Research Group Working Paper 99-1, Department of Computer Sciences, University of Wisconsin, 1999.
- [Dennett, 1996] D.C. Dennett. *Kinds of Mind*. Basic Books, New York, 1996.
- [Giles *et al.*, 1992] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, and G. Z. Sun. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992.
- [Hammer and Tiño, 2003] B. Hammer and P. Tiño. Recurrent neural networks with small weights implement definite memory machines. *Neural Computation*, 15(8):1897–1929, 2003.
- [Jacobsson, 2005] H. Jacobsson. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263, 2005.
- [Popper, 1959] Karl R. Popper. *The Logic of Scientific Discovery*. Hutchinson Education, London, 1959.
- [Sharkey and Jackson, 1995] N. E. Sharkey and S. A. Jackson. An internal report for connectionists. In R. Sun and L. A. Bookman, editors, *Computational Architectures integrating Neural and Symbolic Processes*, pages 223–244. Kluwer, Boston, 1995.
- [Tiño and Vojtek, 1998] P. Tiño and V. Vojtek. Extracting stochastic machines from recurrent neural networks trained on complex symbolic sequences. *Neural Network World*, 8(5):517–530, 1998.

²Not to be confused with Dennett’s Popperian Minds [Dennett, 1996].