Universidad ORT Uruguay

Facultad de Ingeniería

Regular Inference over Recurrent Neural Networks as a Method for Black Box Explainability

Entregado como requisito para la obtención del título de Master en Ingeniería

Franz Mayr - 178236

Tutor: Sergio Yovine

2019

Declaración de Autoría

Yo, Franz Mayr declaro que el trabajo que se presenta en esta obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el Proyecto;
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mi;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Franz Mayr

17-06-2019

Agradecimientos

Dedicado a mi familia y amigos, quienes fueron mi soporte en esta etapa.

Agradezco a mi tutor, Dr. Sergio Yovine, por las largas discusiones que dieron fruto a este trabajo.

Abstract Español

El presente trabajo es una extensión de [1], su objetivo es explorar el problema general de explicar el compartamiento de una red neuronal recurrente (RNN por sus siglas en inglés). El objetivo es construir una representación que mejore el entendimiento humano de las RNN como clasificadores de secuencias, con el propósito de proveer entendimiento sobre el proceso de decisión detrás de la clasificación de una secuencia como positiva o negativa, y a su vez habilitar mayor análisis sobre las mismas como por ejemplo verificación formal basada en autómatas. En concreto, se propone un algoritmo de aprendizaje automático activo para la construcción de un autómata finito determinístico que es aproximadamente correcto respecto a una red neurnal artificial.

Abstract

This work is an extension of [1], it explores the general problem of explaining the behavior of recurrent neural networks (RNN). The goal is to construct a representation which enhances human understanding of an RNN as a sequence classifier, with the purpose of providing insight on the rationale behind the classification of a sequence as positive or negative, but also to enable performing further analyses, such as automata-theoretic formal verification. In particular, an active learning algorithm for constructing a deterministic finite automaton which is approximately correct with respect to an artificial neural network is proposed.

Palabras clave

Redes neuronales recurrentes; Clasificación de secuencias; Automata finito determinístico; Aprendizaje aproximadamente correcto; Extracción de reglas; Inferencia regular; Inteligencia Artificial; Explicabilidad

Key words

Recurrent neural networks; Sequence classification; Deterministic finite automata; Probably approximately correct learning; Rule extraction; Regular inference; Artificial Inteligence; Explainability

Contents

1 Introduction

Artificial intelligence (AI) is a thriving field with many practical applications and active research topics. Intelligent software is developed in order to automate processes, understand speech or images, make diagnoses in medicine and support basic scientific research [2].

As these systems are designed and constructed, there are some quality attributes that guide this processes which are not exactly the same as those of a typical software system. One of them is explainability (or interpretability). This attribute goes beyond the basic proposed metrics that the models are trained to achieve. Indeed, defining how to measure the explainability of a model is still ongoing research work in the field of explainable artificial intelligence.[3, 4]

The purpose of explainable artificial intelligence is to come up with artifacts capable of producing intelligent outcomes together with appropriate rationalizations of them. It means that besides delivering best possible model performance metrics (e.g., accuracy) and computational performance metrics (e.g., algorithmic complexity), they must provide adequate and convincing reasons for effectively justifying the judgment in a human-understandable way [1].

Artificial neural networks (ANN) are the state-of-the-art method for many fields in the area of artificial intelligence [5]. However, ANN are considered to be a rather obscure model [6], meaning that understanding the specifics that were taken into consideration by the model to make a decision is not a trivial task.

Human understanding of the model is crucial in fields such as medicine [7], risk assessment [8], or intrusion detection [9]. From the point of view of explaining the rationale of an outcome, an important issue is that ANN lack an explicit and constructive characterization of their embedded decision-making strategy.

This limitation of ANN explanatory capabilities motivated a large amount of research work aiming at improving ANN explainability. Several approaches to tackle this issue have been identified [10, 11]. In particular, [11] characterizes the *black-*

box model explanation problem. It consists in providing a human-understandable model which is able to mimic the behavior of the ANN [11]. Three classes of approaches to ANN explainability are identified in [10], namely *processing* explanation, which seeks answering *why* a given input leads the ANN to produce a particular outcome; *representation* explanation, which consists in characterizing the internal structure of the ANN; and *explanation-producing* one, which relies on creating ANNs that are easier to explain by design. Another approach consists in allowing a human actor to interact with the learning process. This human-in-the-loop method of addressing explainability, called *glass box* interactive machine-learning approach, is presented in [12].

The need for explainable artificial intelligence is also advocated by DARPA in its XAI project [13], which is one of the first introductions to the concept. DARPA XAI presents another taxonomy of techniques to tackle this problem: *deep explanation*, which consists in modified deep learning techniques to learn explainable features; *explainable models* which means developing techniques to learn more structured, interpretable, causal models; and *model induction* which implies inferring an explainable model from any model as a black box.

In this work a black-box model and processing explanation approach for ANN is followed. If we look at our approach from the optic of the XAI project, we can say that model induction is our way of pursuing explainability.

Our main interest is studying explainability in the context of recurrent neural networks (RNN) trained to solve *sequence* classification problems [14, 15], which appear in many application domains. In the past few years several classes of RNN, e.g., Long-Short Term Memory (LSTM) [16], have been successfully applied for such matter [17, 18, 19, 20, 21, 22, 23, 24].

In this work the study is restricted to binary classification. This problem is a case of *language membership*, where the language of the RNN is the set of sequences classified as positive by the network. The trained RNN hides a model of such sequences which it uses to predict whether a given input sequence belongs to the language. If the language from which the training samples have been drawn is known, the question is how well the RNN learned it [25, 26, 17]. Another, may be more realistic, situation occurs when the target language is unknown. In this case, the question to answer becomes what is the language learned by the RNN, or more precisely, whether it could be characterized operationally instead of denotationally.

Typically, these questions are addressed by looking at the accuracy of the network on a given test set. However, it has been observed that networks trained with millions of samples which exhibit 100% accuracy on very large development

test sets can still incorrectly classify random sequences [27, 24]. Thus, exact convergence on a set of sequences, whatever its size, does not ensure the language of the network is the same as the target language. Hence, in both cases, the question remains whether the language recognized by the network can be explained, even approximately, in some other, human-comprehensible, way.

The goal of this work is to provide means for extracting a constructive representation of the model hidden inside the RNN. To the best of our knowledge, most previous works devoted to model and processing explanation for RNN are either white-box, that is, they look into and/or make assumptions about the network's structure and state, or they are focused on extracting decision trees or rules. A thorough comparison with related work is given in Section 5

Now, when it comes to operationally explaining the dynamical system that produces sequences of events, rules and trees are not expressive enough. In this case, widely used formalisms are automata [28]. This model provides a language-independent mathematical support for studying dynamical systems whose behavior could be understood as sequences corresponding to words of a regular language.

In the automata-theoretic approach, when the system under analysis is a black box, that is, its internal structure (composed of states and transitions) is unknown, the general problem of constructing an automaton that behaves as the black box is called *identification* or *regular inference* [29].

Many times it is not theoretically or practically feasible to solve this problem precisely, in which case, it needs to be solved approximately. That is, rather than exactly identifying the automaton inside the black box, we attempt to find an automaton which is a reasonable approximation with some confidence.

The Probably Approximately Correct (PAC) framework [30] is a general approach to solve problems like the one we are considering here. A *learner*, which attempts to identify the hidden machine inside the black box, can interact with a *teacher*, which has the ability to answer queries about the unknown machine to be learned. For this, the teacher uses an oracle which draws positive and negative samples with some probability distribution. PAC theory is based on a general notion of *concept*. Here, only its application to regular languages, or equivalently, to deterministic finite automata, which are a specific class of concepts is considered. There are several specific problem instances and algorithms to solve this problem, depending on the assumptions that are made regarding how the behavior of the black box is observed, what questions could be asked, how the answers to these questions could be used to build an automaton, etc. The reader is referred to [31, 29] for a thorough review.

In this context, two general settings can be distinguished, namely *passive* or *active* learning. The former consists in learning a language from a set of given (chosen by the teacher) positive and/or negative examples [32]. It has been shown in [33] that this problem is *NP-complete*. In the latter, the learner is given the ability to draw examples and to ask membership queries to the teacher. A well known algorithm in this category is Angluin's L^* [34]. L^* is *polynomial* in the number of states of the minimal deterministic finite automaton (DFA) and the maximum length of any sequence exhibited by the teacher.

The relationship between automata and RNN has been thoroughly studied: [25] presents mechanisms for programming an RNN that can correctly classify strings of arbitrary length belonging to a given regular language; [26] discuss an algorithm for extracting the finite state automaton of second-order recurrent neural networks; [24] look at this problem in a white-box setting. Therefore, according to [11], a black-box model explanation approach, that is, using regular inference algorithms that do not rely on the RNN structure and weights is a question that has not been addressed so far.

Of course, one may argue that an automaton could be directly learned from the dataset used to train the network. However, this approach has several drawbacks. First, passive learning is NP-complete [33]. Second, it has been shown that RNN such as LSTM, are much better learners, as they learn faster and generalize better. Besides, they are able to learn languages beyond regular ones. Third, the training dataset may not be available, in which case the only way to construct an explanation is to query the RNN.

The contribution of this work is an adaptation of Angluin's L^* algorithm that outputs a DFA which approximately behaves like an input RNN whose actual structure is completely unknown. This means that whenever a sequence is recognized by the DFA constructed by the algorithm, it will most likely be classified as positive by the RNN, and vice versa. It is important to stress the fact that the algorithm proposed is completely agnostic of the structure of the RNN.

Outline

In Chap. 2 we precisely present the problem we are going to address and the method used for solving it. In Chap. 3 we discuss the proposed algorithm and a variation of the general PAC framework to analyze its behavior. In Chap. 4 we present the experimental results carried out on several examples which validate the theoretical analyses. In Chap. 5 we compare and contrast our approach with related works. Finally we present the conclusions and future work.

2 Preliminaries

2.1 Learning Regular Languages

2.1.1 Regular Languages

Regular languages can be defined as the ones that can be described by deterministic finite automata (DFA) [35]. DFA are formally defined as a tuple $(Q,\ \Sigma,\ \delta,\ q_0,\ F)$ where:

1. Q is a finite set of states.
2. Σ is a finite set of input symbols,.
3. δ is a transition function that takes as arguments a state and an input symbol and returns a state.
4. q_0 is a start state, belonging to Q.
5. F is a set of final or accepting states, F being a subset of Q.

The Chomsky hierarchy defines these languages as the languages that are generated by Type-3 grammars (regular grammars) [36].

From the description in [35], the key point is that these models characterize the languages, meaning they provide a constructive way of describing them, and recognizing their elements (that is, checking whether a sequence of symbols does belong to the language).

A simple example of a regular language is presented in Figure 2.1. This language is described by the regular expresion $(ab)^*$, with:

1. $Q = \{0, 1, 2\}$.

2. $\Sigma = \{a, b\}$ and λ being the empty sequence.

3. δ as presented graphically in the figure or tabularly in Table 2.1.

4. $q_0 = 0$ (indicated with an incoming arrow).

5. $F = \{0\}$ (indicated with a double circle).

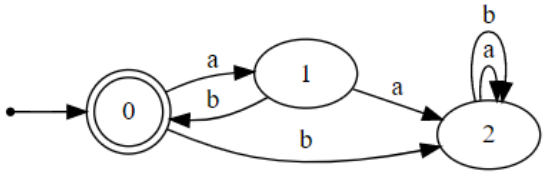


Figure 2.1: Example of automaton

δ	**a**	**b**
0	1	2
1	2	0
2	2	2

Table 2.1: Table of transition function δ of automaton in Fig. 2.1

2.1.2 Grammatical Inference

Grammatical Inference is defined as the problem of inducing, learning or inferring grammars. It is a field with connections to a series of disciplines such as bio-informatics, computational linguistics and pattern recognition [29]. The goal of this field is to infer grammars given some information about the languages, and, as grammars are constructive models, they present insight and generalization over the words belonging to the language, something that most state of the art neural models lack.

There are two settings that the learning processes could adopt, and those are active learning, and passive learning.

Passive learning consists in learning a language from a set of given positive and/or negative examples [32]. It has been shown in [33] that finding a minimal DFA that is consistent with a given arbitrary set of sequences is *NP-complete*.

In the active learning setting, the learner is given the ability to draw examples and to ask membership queries to the teacher. A well known algorithm in this category is Angluin's L^* [34]. L^* is *polynomial* in the number of states of the minimal deterministic finite automaton (DFA) and the maximum length of any sequence exhibited by the teacher.

2.1.3 L^*

A well known algorithm in the category of active learning is Angluin's L^* [34].

L^* constructs a DFA by interacting with a Minimum Adequate Teacher (MAT) that exposes two operations: a *membership query* (MQ), that is a boolean response if a given sequence is accepted by the language known by the teacher, and an *equivalence test* (EQ), that is a function that compares the target language and the inferred one, if they are equivalent the test returns true, if not it returns a counterexample (a word belonging to one of the languages but not the other).

The way the algorithm achieves the learning is as follows. It builds a table of observations by interacting with the MAT. This table is used to keep track of which words are and are not accepted by the target language. The construction of this table is done in an iterative way by asking the teacher membership queries through the Membership Oracle (MQ) of different words in order to fill the Observation Table (OT).

The information that is in the observation table has three characteristics. A nonempty finite prefix-closed set of strings (every prefix of every member is also a member of the set), a nonempty finite suffix-closed set of strings (every suffix of every member is also a member of the set), and a finite function that maps a string to either 1 or 0 if it is a member of our target language or not respectively.

The observation table is composed by two sets of rows: the 'upper' rows (or top part, that we will call **RED** following De la Higuera's notation [29]), that represent the elements of the prefix-closed set of strings mentioned earlier, and the 'lower' rows (or bottom part, that we will call **BLUE**), which represent the same elements of this set but concatenated with the set of letters in the language alphabet. On the other hand, columns represent a suffix-closed set of strings, and each cell represents the membership relationship, both also mentioned earlier. An example of the observation table is presented in Table 2.2b.

The observation table is first initialized by building one **RED** row (Ffor the empty word λ) and one **BLUE** row for each symbol in the alphabet Σ (length-one words). Then the iterative process begins.

In order to be able to make sense out of the table, it needs to comply with two properties. First of all, it needs to be closed. The table is considered closed if, for every row in the bottom part of the table, there is an equal row in the top part. The second property is consistency. A table is considered consistent if for every pair of rows in the top part of the table (**RED**) with the same values (same order of 0s and 1s), then all pairs of extensions with the same letter of the alphabet must have the same row in the table. Precisely, a table is consistent if for every different row in **RED**, for every symbol x in Σ if $OT[v] = OT[w]$ then $OT[v.x] = OT[w.x]$.

If the table is not closed, the algorithm moves to the **RED** part a row in the **BLUE** part that does not have an equal row in the **RED** part and adds to the **BLUE** set all the rows corresponding to the extensions of its associated word with every letter of the alphabet.

To make it consistent, the algorithm expands the original set of suffixes with the letter that makes their corresponding extensions different (an $x \in \Sigma$ such that $OT[v] = OT[w]$ but $OT[v.x] \neq OT[w.x]$). This is done in order to differentiate between the two words that had the same row values.

Once the table is closed and consistent, the algorithm proceeds to construct the conjectured DFA and then asks the oracle whether it is equivalent to the target one. If the answer is yes, it terminates and returns the learned DFA. If the answer is no, then it receives a counterexample that proves the DFA is wrong, and it proceeds to extend the observation table with this new counter example. This extension is done by adding every prefix of the counterexample to **RED**, and for each prefix its concatenation with every symbol in Σ to **BLUE** (given that the concatenation is not a prefix).

2.1.3.1 An L^* run

Let us take as an example the regular language presented in Figure 2.1, described by the regular expression $(ab)^*$.

First, the algorithm constructs the table as presented in 2.2a. As the table is not closed (not every row in **BLUE** has a representation in **RED**), the algorithm proceeds to close it. To do that, one of the elements in **BLUE** that has not a

representative in **RED** is selected, for example a, and moves it to **RED**, adding to **BLUE** its concatenation to every symbol (aa and ab, then the holes are filled. The resulting table can be seen in Table 2.2b.

As the observation table is now closed (the previous step solved this problem) and consistent an automaton can be built.

To build an automaton out of the table, the states are represented by every unique row in **RED**, the final states are those corresponding to the rows w where $OT[w][\lambda] = 1$, and rejecting states are those rows v where $OT[v][\lambda] = 0$. Finally the transition function is defined as: $\delta(q_v, a) = w$ if $OT[va] = OT[w]$. The resulting automaton is presented in Figure 2.2.

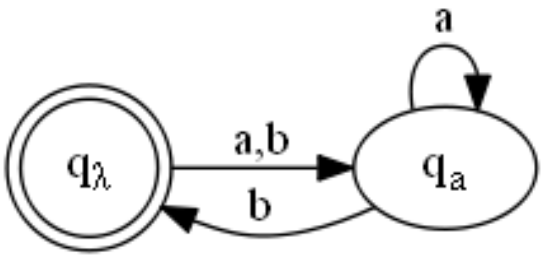


Figure 2.2: First proposed automaton in an L^*example run.

This automaton is then presented to the teacher via the **EQ**, which can be implemented by the table-filling algorithm [35]. This query results negative, as the regular language that the conjectured automaton represents is not the same as the target one. Let us suppose that the counterexample returned by the teacher is 'bb'.

Now, the learner proceeds to process the counterexample. This is done by adding the counterexample and all its prefixes to **RED**, and at the same time adding for each prefix v and for all symbol x, $v.x$ to **BLUE**, given that $v.x$ is not a prefix of the counterexample. Then holes are filled, resulting in the Table 2.2c

The table remains closed, however it is not consistent, as two **RED** rows have different resulting rows if they are added a symbol. To be concrete, $OT[a] = OT[b]$, however $OT[ab] \neq OT[bb]$). This can be informally interpreted as 'they seem to be the same state in the table, however they are not', so they have to be separated. This separation is achieved by adding the symbol that makes them differ to the columns of the observation table (in this case symbol b). The symbol is added, holes are filled, the result is Table 2.2d.

OT_0	λ
λ	1
a	0
b	0

(a)

OT_1	λ
λ	1
a	0
b	0
aa	0
ab	1

(b)

OT_2	λ
λ	1
a	0
b	0
bb	0
aa	0
ab	1
ba	0
bba	0
bbb	0

(c)

OT_3	λ	b
λ	1	0
a	0	1
b	0	0
bb	0	0
aa	0	0
ab	1	0
ba	0	0
bba	0	0
bbb	0	0

(d)

Table 2.2: Observation tables during an L^*example run.

The last table is closed and consistent, the conjectured automaton is finally equivalent to the target one, so **EQ** outputs true and L^* finishes and the DFA present in Figure 2.3, which is equivalent to the target one, is returned.

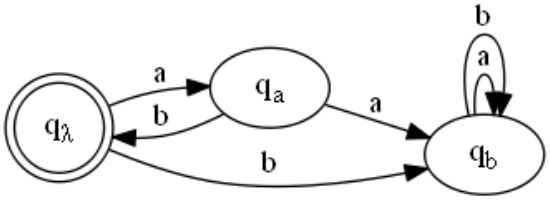


Figure 2.3: Output automaton in an L^* example run.

2.1.4 Probably Approximately Correct learning

The previous algorithm asumes that the teacher has a means of comparing the target DFA and the proposed one in an exact way, which is not always the case. In her paper [34] Angluin proposes to resort to Valiant's Probably Approximately Correct (PAC) framework [30, 31] in the scenarios where there is no access to a deterministic model comparison.

Since we are interested in learning languages, we restrict ourselves to briefly describing the PAC-learning setting for languages.

Let $\mathcal{D}$ be an unknown distribution over Σ^* and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$. The *symmetric difference* between $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted $\mathcal{L}_1 \oplus \mathcal{L}_2$, is the set of sequences that belong to only one of the languages, that is, $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}_1 \setminus \mathcal{L}_2 \cup \mathcal{L}_2 \setminus \mathcal{L}_1$.

The *prediction error* of $\mathcal{L}_1$ with respect to $\mathcal{L}_2$ is the probability of a sequence to belong to their symmetric difference, denoted $\mathbb{P}_{\mathcal{D}}(\mathcal{L}_1 \oplus \mathcal{L}_2)$. Given $\epsilon \in (0, 1)$, we say that $\mathcal{L}_1$ is ϵ-*approximately correct* with respect to $\mathcal{L}_2$ if $\mathbb{P}_{\mathcal{D}}(\mathcal{L}_1 \oplus \mathcal{L}_2) < \epsilon$.

The *oracle* $\mathbf{EX}_{\mathcal{D}}(\mathcal{L}_1)$ draws an *example* sequence $x \in \Sigma^*$ following distribution $\mathcal{D}$, and tags it as *positive* or *negative* according to whether it belongs to $\mathcal{L}_1$ or not. Calls to **EX** are independent of each other.

A *PAC-learning algorithm* takes as input an *approximation* parameter $\epsilon \in (0, 1)$, a *confidence* parameter $\delta \in (0, 1)$, *target* language $\mathcal{L}_t$ and oracle $\mathbf{EX}_{\mathcal{D}}(\mathcal{L}_t)$, and if it terminates, it outputs a language $\mathcal{L}_o$ such that $\mathcal{L}_o$ is ϵ-approximately correct with respect to $\mathcal{L}_t$ with probability at least $1 - \delta$.

A PAC-learning algorithm can also be equipped with an *approximate equivalence* test **EQ** which checks a candidate output $\mathcal{L}_o$ against the target language $\mathcal{L}_t$ using a *sufficiently large* sample of tagged sequences S generated by **EX**. If the sample is such that for every $x \in S$, $x \in \mathcal{L}_t \iff x \in \mathcal{L}_o$, the algorithm successfully stops and outputs $\mathcal{L}_o$. Otherwise, it picks any sequence in $S \cap (\mathcal{L}_o \oplus \mathcal{L}_t)$ as *counterexample* and continues.

The algorithm may also be allowed to call directly a *membership* oracle **MQ**, such that $\mathbf{MQ}(x, \mathcal{L}_t)$ is true if and only if $x \in \mathcal{L}_t$. Notice that the **EX** oracle may also call **MQ** to tag sequences.

A *distribution-free* algorithm is one that works for every $\mathcal{D}$. Hereinafter, we will focus on distribution-free algorithms, so we will omit $\mathcal{D}$.

2.1.4.1 PAC-based L^*

In this setting, the L^* learner is the same, however, the teacher resorts to a statistical test in order to compare the languages. This test is explained below.

Given a DFA $\mathcal{A}$, we use $\mathcal{L}(\mathcal{A})$ to denote its language, that is, the set of sequences accepted by $\mathcal{A}$. We denote $\mathcal{A}_t$ and $\mathcal{A}_o$ the target and output automata, respectively. The symmetric difference between A_o and A_t, denoted $A_o \oplus \mathcal{A}_t$, is defined as $\mathcal{L}_o \oplus \mathcal{L}_t$. We say that $\mathcal{A}_o$ ϵ-approximates $\mathcal{A}_t$ if $\mathcal{L}_o$ ϵ-approximates $\mathcal{L}_t$.

L^* uses **EQ** and **MQ**. Each time **EQ** is called, it must draw a sample of a

size large enough to ensure a *total* confidence of the algorithm of at least $1 - \delta$. That is, whenever the statistical test is passed, it is possible to conclude that the candidate output is ϵ-approximately correct with confidence at least $1 - \delta$.

Say **EQ** is called at iteration i. In order to guarantee the aforementioned property, a sample S_i of size r_i is drawn, where:

$$r_i = \left\lceil \frac{1}{\epsilon} \left(i \ln 2 - \ln \delta \right) \right\rceil \tag{2.1}$$

This ensures that the probability of the output automaton $\mathcal{A}_o$ *not* being ϵ-approximately correct with respect to $\mathcal{A}_t$ when *all* sequences in a sample S_i pass the **EQ** test, i.e., $S_i \cap (\mathcal{A}_o \oplus \mathcal{A}_t) = \emptyset$, is *at most* δ, that is:

$$\sum_{i>0} \mathbb{P}(S_i \cap (\mathcal{A}_o \oplus \mathcal{A}_t) = \emptyset \mid \mathbb{P}(\mathcal{A}_o \oplus \mathcal{A}_t) > \epsilon) < \sum_{i>0} (1-\epsilon)^{r_i} < \sum_{i>0} 2^{-i}\delta < \delta$$

Remark

It is worth noticing that from the point of view of statistical hypothesis testing, a sample S_i that passes the test, gives a confidence of at least $1 - 2^{-i}\delta$.

2.2 Recurrent Neural Networks

Recurrent neural networks, or RNNs [37], are a family of neural networks for processing sequential data.

In other words, this family of neural networks is specialized for processing a sequence of values $x^{(1)}, ..., x^{(\tau)}$, for any variable τ. The RNN processes information by using its internal continuous state space as an implicit, holistic, memory of past input patterns [38].

One may also look at a RNN as a dynamical system, whose state $s^{(t)}$ depends on its previous state $s^{(t-1)}$, an input value $x^{(t)}$ and some parameters θ that are not dependent on time, in which case, the general equation is the one described in Eq. 2.2:

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta), \tag{2.2}$$

Goodfellow et al [2] present some examples of important design patterns for recurrent neural networks:

1. Recurrent networks that produce an output ($o^{(t)}$) at each time step and have recurrent connections between hidden units.

2. Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step

3. Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output

In this work, we will focus on the latter, which is presented in a simplified version in Figure 2.4 borrowed from [2]. The equations asociated to the image are:

$$s^{(t)} = h^{(t)} = g_h \left(W h^{(t-1)} + U x^{(t)} \right) \tag{2.3}$$

$$o^{(\tau)} = g_o \left(V h^{(\tau)} \right) \tag{2.4}$$

where g_h and g_o are *activation* functions, such as *reLU* and *softmax*, and L is a *loss* function to be minimized, such as the *cross-entropy* $\mathbb{E}(y \log o)$.

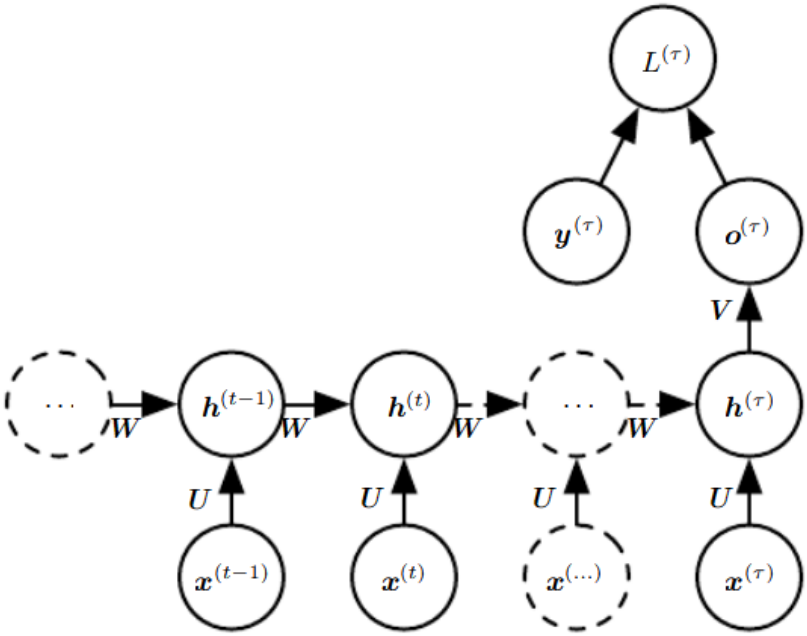


Figure 2.4: Time-unfolded recurrent neural network with a single output at the end of the sequence, borrowed from [2].

Long Short-Term Memory RNNs

With regards to RNNs, more complex architectures have been developed to tackle some of the main problems related to learning long time dependencies in sequences, one of these is the Long Short-Term Memory architecture [16].

They solve the shortcomings of the vanishing gradient problem [39], an optimization problem where models fail to converge due to small gradients during training time, through the training of special units.

Minor details of this architecture will not be described as it is out of the scope of the present work.

2.3 Problem Statement

Let $\mathcal{U} \subseteq \Sigma^*$ be some *unknown* language over an alphabet Σ of symbols, and $\mathcal{S} \subseteq \Sigma^*$ be a sample set such that it contains positive and negative sequences of $\mathcal{U}$. That is, there are sequences in $\mathcal{S}$ which belong to $\mathcal{U}$ and others that do not.

The *language* of an RNN $\mathcal{N}$, denoted $\mathcal{L}(\mathcal{N})$, is the set of sequences classified as positive by $\mathcal{N}$. Suppose $\mathcal{N}$ is obtained by training it with a sample set $\mathcal{S}$, and then used to predict whether a sequence $u \in \Sigma^*$ does belong to $\mathcal{U}$. In other words, the unknown language $\mathcal{U}$ is considered to be somehow approximated by $\mathcal{L}(\mathcal{N})$, that is, with high probability $x \in \mathcal{L}(N) \iff x \in \mathcal{U}$.

But, what is the actual language $\mathcal{L}(\mathcal{N})$ learned by $\mathcal{N}$? Is it a regular language? That is, could it be expressed by a deterministic finite automaton? Is it possible to approximate it somehow with a regular language? The interest of having an automaton-based, either precise or approximated, characterization of $\mathcal{L}(\mathcal{N})$, allows us to explain the answers of $\mathcal{N}$, while providing insight on the unknown language $\mathcal{U}$. This approach enhances human understanding because of two main characteristics: it enables a visual and symbolic representation, which is aligned with the Michalski's comprehensibility postulate [40], and it also allows performing further analyses, such as automata-theoretic formal verification [41].

3 PAC-learning for RNN

We address the following problem: given an RNN $\mathcal{N}$, is it possible to build a DFA $\mathcal{A}$, such that $\mathcal{L}(\mathcal{A})$ is ϵ-approximately correct with respect to $\mathcal{L}(N)$?

The basic idea to solve this problem is to use L^* as follows. The **MQ** oracle consists in querying $\mathcal{N}$ itself. The **EQ** oracle consists in drawing a sample set S_i with size r_i as defined in equation (2.1) and checking whether $\mathcal{N}$ and the candidate automaton $\mathcal{A}_i$ completely agree in S_i, that is, $S_i \cap (\mathcal{A}_i \oplus \mathcal{N})$ is empty.

The results reviewed in the previous section entail that if L^* terminates, it will output a DFA $\mathcal{A}$ which is an ϵ-approximation of $\mathcal{N}$ with probability at least $1 - \delta$. Moreover, L^* is proven to terminate *provided $\mathcal{L}(\mathcal{N})$ is a regular language*. However, since RNN are strictly more expressive than DFA, as they are proven to be Turing complete [42, 43], there is no guarantee that L^* will eventually terminate; there may not exist a DFA $\mathcal{A}$ with the same language as $\mathcal{N}$. In other words, there is no upper bound $n_{\mathcal{N}}$ such that L^* will terminate in at most $n_{\mathcal{N}}$ iterations for every target RNN $\mathcal{N}$. Therefore, it may happen that for every i the call to **EQ** fails, that is, $S_i \cap (\mathcal{A}_i \oplus \mathcal{N}) \neq \emptyset$.

The rest of the chapter is focused in presenting a solution to this problem.

3.1 Bounded-L^*

To cope with the aforementioned situation, we resort to imposing a bound to the number of iterations of L^*. Obviously, a direct way of doing it would be to just fix an arbitrary upper bound to the number of iterations. Instead, we propose to constrain the maximum number of states of the automaton to be learned and to restrict the length of the sequences used to call **MQ**. The latter is usually called the *query length*. Typically, these two measures are used to determine the complexity of a PAC-learning algorithm [44].

3.1.1 Algorithm

Similarly to the description of L^* presented in [29] and sketched in Chapter 2, the algorithm Bounded-L^*(Algorithm 1) can be described as follows:

Algorithm 1: Bounded-L^*

```
   Input  : MaxQueryLength, MaxStates, ϵ, δ
   Output: DFA A
 1 Lstar-Initialise;
 2 repeat
 3 |  while OT is not closed or not consistent do
 4 |  |  if OT is not closed then
 5 |  |  |  OT, QueryLengthExceeded ← Lstar-Close(OT);
 6 |  |  end
 7 |  |  if OT is not consistent then
 8 |  |  |  OT, QueryLengthExceeded ← Lstar-Consistent(OT);
 9 |  |  end
10 |  end
11 |  if not QueryLengthExceeded then
12 |  |  A← Lstar-BuildAutomaton(OT);
13 |  |  Answer ← EQ(A);
14 |  |  MaxStatesExceeded ← STATES(A)>MaxStates;
15 |  |  if Answer ≠ Yes and not MaxStatesExceeded then
16 |  |  |  OT ← Lstar-UseEQ(OT, Answer);
17 |  |  end
18 |  end
19 |  BoundReached ← QueryLengthExceeded or MaxStatesExceeded;
20 until Answer = Yes or BoundReached;
21 return A;
```

The observation table OT is initialised by Lstar-Initialise in the same manner that it is for L^*. This step consists in building the structure of the observation table OT as proposed by Angluin. Then the construction of hypotheses begins.

If OT is not *closed* an extra row is added by the Lstar-Close procedure. If OT is *inconsistent*, an extra column is added by the Lstar-Consistent procedure. Both procedures call **MQ** to fill the holes in the observation table. The length of these queries may exceed the maximum query length, in which case the QueryLength-Exceeded flag is set to true.

When the table is closed and consistent, and in the case that the maximum query length was not exceeded, an equivalence query **EQ** is made and the number of states of the automaton is compared to the maximum number of states bound. If **EQ** is unsuccessful and the maximum number of states was not reached, new rows are added by Lstar-UseEQ, using the counterexample contained in *Answer*.

Finally, if the hypothesis passes the test or one of the bounds was reached, the algorithm stops and returns the last proposed automaton.

In Fig. 3.1 a diagram of the teacher-learner setting is presented.

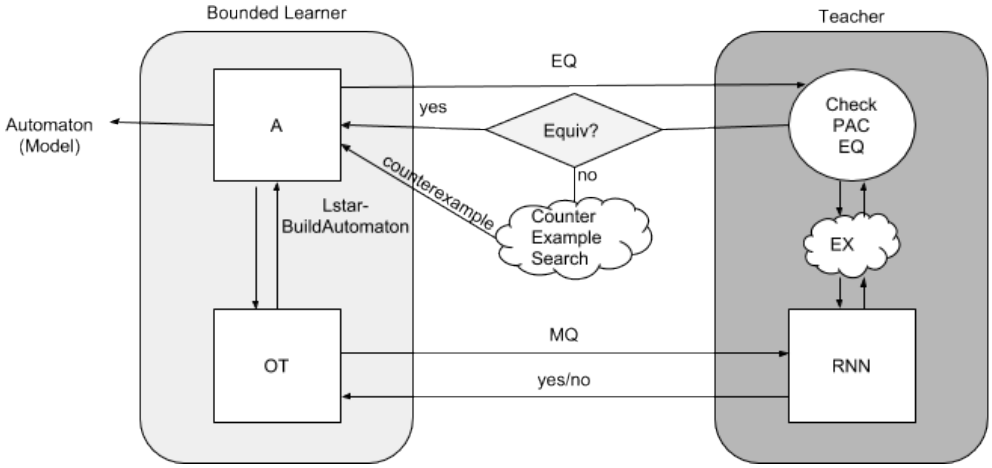


Figure 3.1: Experimental setting diagram

3.1.2 Analysis

Bounded-L^* will either terminate with a successful **EQ** or when a bound (either the maximum number of states or query length) is exceeded. In the former case, the output automaton $\mathcal{A}$ is proven to be an ϵ-approximation of $\mathcal{N}$ with probability at least $1 - \delta$ by Angluin's results. In the latter case, the output $\mathcal{A}$ of the algorithm will be the *last automaton proposed by the learner*. $\mathcal{A}$ may not be an ϵ-approximation with confidence at least $1 - \delta$, because $\mathcal{A}$ failed to pass the last statistical equivalence test **EQ**. However, the result of such test carries statistical value about the relationship between $\mathcal{A}$ and $\mathcal{N}$. The question is, what could indeed be said about this automaton?

Assume at iteration i **EQ** fails and the number of states of $\mathcal{A}_i$ is greater than

or equal to the maximum number of states, or at the next iteration $i+1$, **MQ** fails because the maximum query length is exceeded. This means that $\mathcal{A}_i$ and $\mathcal{N}$ disagree in, say, $k > 0$, of the r_i sequences of S_i. In other words, there are k sequences in S_i which indeed belong to the symmetric difference $\mathcal{A}_i \oplus \mathcal{N}$.

Confidence parameter.

Let $p \in (0,1)$ be the actual probability of a sequence to be in $\mathcal{A}_i \oplus \mathcal{N}$ and K_{r_i} the random variable defined as the number of sequences in $\mathcal{A}_i \oplus \mathcal{N}$ in a sample of size r_i. Then, the probability of $K_{r_i} = k$ is:

$$\mathbb{P}(K_{r_i} = k) = \binom{r_i}{k}(1-p)^{r_i-k}p^k$$

Let us first set as our hypothesis that $\mathcal{A}_i$ is an ϵ-approximation of $\mathcal{N}$. Can we accept this hypothesis when $K_{r_i} = k$ with confidence at least $1-\delta'$, for some $\delta' \in (0,1)$? In other words, is there a δ' such that the probability of $K_{r_i} = k$ is smaller than δ' when $p > \epsilon$? Suppose $p > \epsilon$. Then, it follows that:

$$\mathbb{P}(K_{r_i} = k \mid p > \epsilon) = \binom{r_i}{k}(1-p)^{r_i-k}p^k < \binom{r_i}{k}(1-\epsilon)^{r_i-k} < \binom{r_i}{k}e^{-\epsilon(r_i-k)}$$

Therefore, if the following condition holds

$$\binom{r_i}{k}e^{-\epsilon(r_i-k)} < \delta' \tag{3.1}$$

we have that $\mathbb{P}(K_{r_i} = k \mid p > \epsilon) < \delta'$. That is, the probability of incorrectly accepting the hypothesis with k discrepancies in sample S_i of size r_i is smaller than δ'. Then, we could accept the hypothesis with a confidence of at least $1-\delta'$.

The left-hand-side term in condition (3.1) gives us a lower bound δ_i^* for the confidence parameter such that we could accept the hypothesis with probability at least $1-\delta'$, for every $\delta' > \delta_i^*$:

$$\delta_i^* = \binom{r_i}{k}e^{-\epsilon(r_i-k)} \tag{3.2}$$

A major problem, however, is that δ_i^* may be greater than 1, and so no $\delta' \in (0,1)$ exists, or it may be just too large, compared to the desired δ, to provide an acceptable level of confidence for the test.

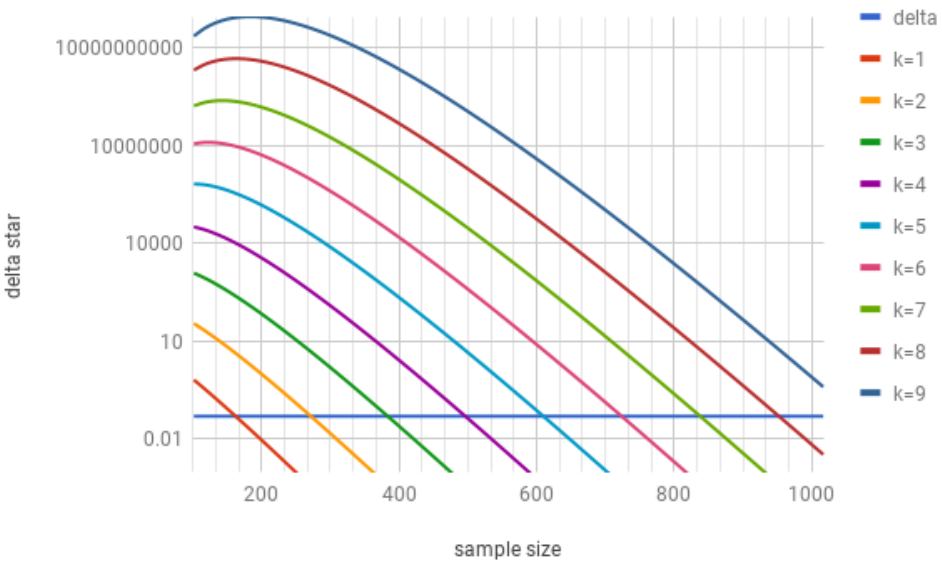


Figure 3.2: Values of δ_i^* in log scale as function of r_i, $k \in [1, 9]$, $i \in [3, 70)$

Fig. 3.2 shows δ_i^*, in log scale, for $\epsilon = 0.05$, $k \in [1, 9]$ and r_i computed using equation (2.1), and compares it with a desired $\delta = 0.05$. We see that as k increases, larger values of r_i, or equivalently, more iterations, are needed to get a value of δ_i^* smaller than δ (horizontal line).

Actually, for fixed k and $\epsilon \in (0, 1)$, δ_i^* tends to 0 as r_i tends to ∞. In other words, there is a large enough sample size for which it is possible to make δ_i^* smaller than any desired confidence parameter δ.

Approximation parameter.

An alternative would be to look at the approximation parameter ϵ, rather than the confidence parameter δ. In this case, we set as our hypothesis that $\mathcal{A}_i$ is an ϵ'-approximation of $\mathcal{N}$, for some $\epsilon' \in (0, 1)$. Is the probability of accepting this hypothesis with the test tolerating k discrepancies, when the hypothesis is actually false, smaller than δ? That is, we are asking whether the following condition holds for ϵ':

$$\mathbb{P}(K_{r_i} = k \mid p > \epsilon') < \binom{r_i}{k} e^{-\epsilon'(r_i - k)} < \delta$$

Now, we could determine a lower bound ϵ_i^*, such that this condition holds for every $\epsilon' > \epsilon_i^*$, in which case we could conclude that $\mathcal{A}_i$ is an ϵ'-approximation of $\mathcal{N}$, with

confidence at least $1 - \delta$. Provided $r_i - k \neq 0$, we have:

$$\epsilon_i^* = \frac{1}{r_i - k}\left(\ln \binom{r_i}{k} - \ln \delta\right)$$

Fig. 3.3 shows ϵ_i^* for $\delta = 0.05$, $k \in [1, 9]$ and r_i computed using equation (2.1), and compares it with a desired $\epsilon = 0.05$. We see that as k increases, larger samples, i.e., more iterations, are needed to get a value smaller than ϵ (horizontal line).

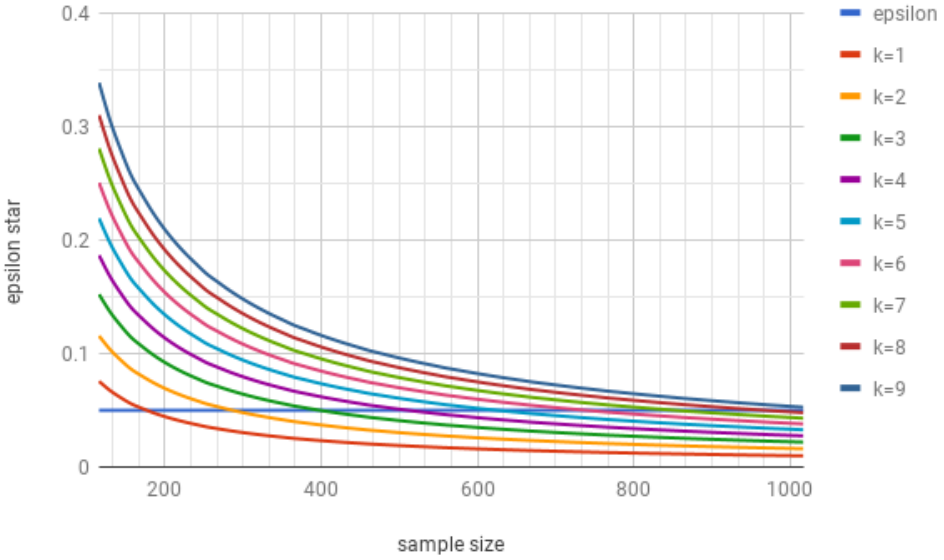


Figure 3.3: Values of ϵ_i^* as function of r_i, $k \in [1, 9]$, $i \in [3, 70)$

Nevertheless, for fixed k and $\delta \in (0, 1)$, ϵ_i^* tends to 0 as r_i tends to ∞. In other words, there is a large enough sample size for which it is possible to make ϵ_i^* smaller than any desired approximation parameter ϵ.

Number of discrepancies and sample size.

Actually, we could also search for the largest number k^* of discrepancies which the **EQ** test could cope with for given ϵ, δ and whichever sample size r, or the smallest sample size r^* for fixed ϵ, δ and number of discrepancies k, independently of the number i of iterations.

The values k^* and r^* could be obtained by solving the following equation for

k and r, respectively:

$$\ln \binom{r}{k} - \epsilon(r-k) - \ln \delta = 0 \tag{3.3}$$

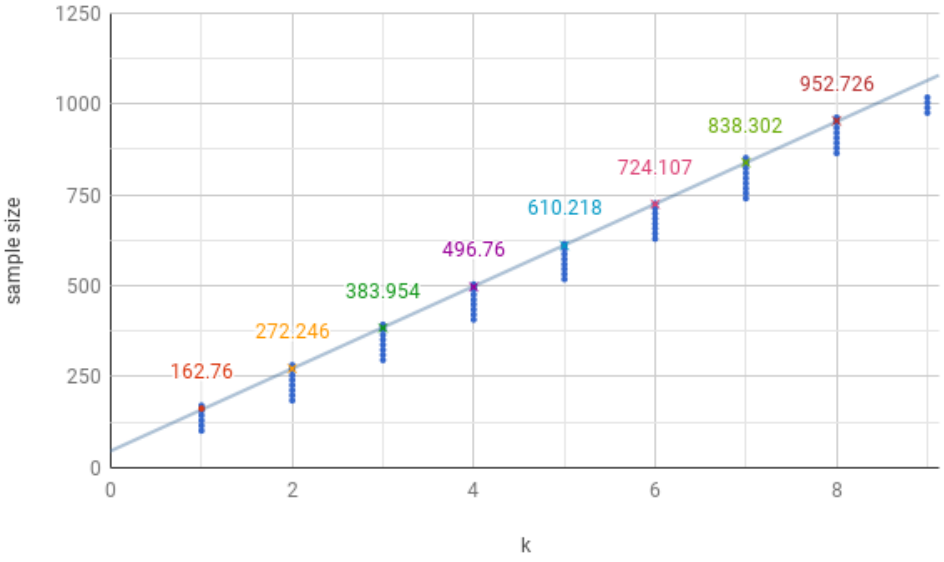


Figure 3.4: Comparison between r_i, r^* and k

Fig. 3.4 plots the relationship between $k \in [1, 9]$, r_i computed using equation (2.1), and $r^*(k)$, where $r^*(k)$ denotes the value of r^* for the given k. Each point in the vertical dotted segments corresponds to a value of sample size r_i. For a given value of k, we plot all values of r_i up to the first one which becomes greater than $r^*(k)$. For each value of k, the value of $r^*(k)$ is shown in numbers.

Whenever r_i is greater than $r^*(k)$, if **EQ** yields at most k discrepancies at iteration i then $\mathcal{A}_i$ could be accepted as being ϵ-approximately correct with respect to $\mathcal{N}$ with confidence at least $1 - \delta$.

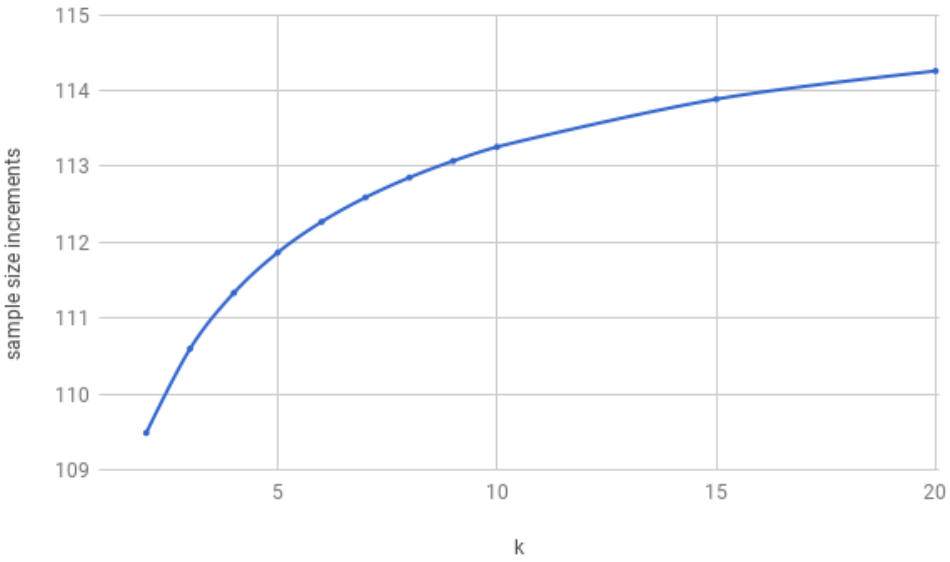


Figure 3.5: Increments of r^* as function of k

The diagonal line in Fig. 3.4 is a linear regression that shows the evolution of r^* as a function of k. Notice that the value of r^* seems to increase linearly with k. However, if we look at the increments, we observe that this is not the case. As Fig. 3.5 shows, they most likely exhibit a log-like growth.

Sample size revisited.

The previous observations suggest that it could be possible to cope with an a-priori given number k of acceptable discrepancies in the **EQ** test by taking larger samples. This could be done by revisiting the formula (2.1) to compute sample sizes by introducing k as parameter of the algorithm.

Following Angluin's approach, let us take δ_i to be $2^{-i}\delta$. We want to ensure:

$$\mathbb{P}(K_{r_i} = k \mid \mathbb{P}(\mathcal{A}_o \oplus \mathcal{A}_t) > \epsilon) < \binom{r_i}{k} e^{-\epsilon(r_i - k)} < 2^{-i}\delta$$

Hence, the smallest such r_i is:

$$r_i = \arg\min_{r \in \mathbb{N}} \left\{ \ln \binom{r}{k} - \epsilon(r-k) + i \ln 2 - \ln \delta < 0 \right\} \tag{3.4}$$

Notice that for $k = 0$, this gives the same sample size as in equation (2.1).

Now, with sample size r_i, we can ensure a total confidence of $1 - \delta$:

$$\sum_{i>0} \mathbb{P}(K_{r_i} = k \mid \mathbb{P}(\mathcal{A}_o \oplus \mathcal{A}_t) > \epsilon) < \sum_{i>0} 2^{-i}\delta < \delta$$

Although computing the sample size at each iteration using equation (3.4) does allow us to cope with up to a given number of discrepancies k in the **EQ** test, it does not ensure termination. Moreover, solving equation (3.4) is computationally expensive, and changing the **EQ** test so as to accepting at most k divergences in a set of r_i samples guarantees the same confidence and approximation as passing the standard zero-divergence test proposed in PAC. Hence, in this work we compute r_i as in equation (2.1) and then analyze the values of δ_i^* and ϵ_i^* that result *after* stopping Bounded-$L^*$when a complexity constraint has been reached.

Remark

One could argue that this analysis may be replaced by running the algorithm with less restrictive bounds. The main issue here is that the target concept (the language of the RNN) may not be in the hypothesis space (regular languages). Thus, there is no guarantee a hypothesis exists for which the PAC condition holds for whichever ϵ and δ. So, even if the user fixed looser parameters, there is no guarantee the algorithm ends up producing a PAC-conforming DFA.

4 Experimental results

We implemented Bounded-L^* and applied it to several examples. In the experiments we used LSTM networks.

Two-phase early stopping was used to train the LSTM, with an 80-20% random split for train-test of randomly generated datasets. For evaluating the percentage of sample sequences in the symmetric difference we used 20 randomly generated datasets. The LSTM were trained with sample datasets from known automata as a way of validating the approach. However it is important to remark that in real application scenarios such automata are unknown, or the dataset may not come from a regular language.

4.1 Example 1

Let us consider the language $(a + b)^*a + \lambda$, with $\Sigma = \{a, b\}$ and λ being the empty sequence (Fig. 4.1). We performed 100 runs of the algorithm with different values for the ϵ and δ parameters. For each run Bounded-L^* terminated normally and obtained a DFA that was an ϵ-approximation of the neural network with confidence at least $1 - \delta$. Actually, every learned automata was equivalent to the original automaton and had no differences in the evaluation of the test datasets with regards to the neural network.

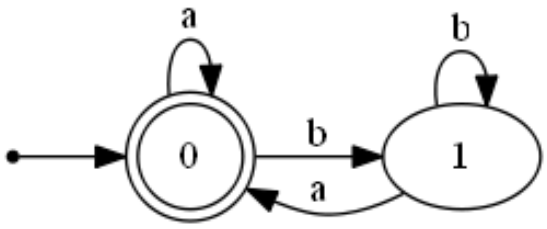


Figure 4.1: Automaton of Exa. 4.1

The previous experiment illustrates that when the neural network is well trained, meaning that it exhibits zero error with respect to all test datasets, the learner will, with high probability, learn the original automaton, which is unknown to both teacher and learner. This case would be the equivalent to using the automaton as the **MQ** oracle, falling in the setting of PAC based L^*. This situation rarely happens in reality, as neural networks, or any model, are never trained to perfectly fit the data. Therefore we are interested in testing the approach with neural networks that do not perfectly characterize the data.

4.2 Example 2

Consider the DFA shown in Fig. 4.2, borrowed from [25]. The training set contained 160K sequences of variable length up to a maximum length of 10. The error measured on another randomly generated sample test set of 16K sequences between the trained LSTM and the DFA was 0.4296. That is, the LSTM does not perform very well: what language did it actually learn?

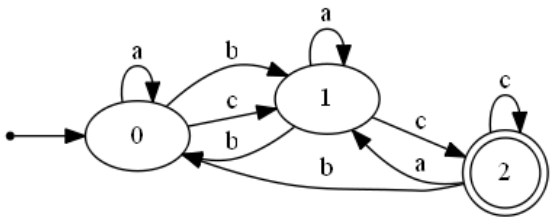


Figure 4.2: Automaton of Exa. 4.2

Bounded-L^* was executed 20 times with $\epsilon = \delta = 0.05$ and a bound of 10 on the number of states. All runs reached the bound and the automaton of the last iteration was the one with smallest error of all iterations.

Fig. 4.3 summarizes the results obtained. It can be observed that for each run of the experiment, not always the same automaton is reached due to the random nature of the **EQ** oracle sample picking. In the 20 runs, 6 different DFA were produced, identified with letters a to f, with a number of states between 11 and 13 (indicated in parenthesis).

For each automaton, the measured error on the test set is obviously the same. However, ϵ_i^* and δ_i^* may differ, because they depend on the number of iterations and on the number of discrepancies on the randomly generated sample sets used

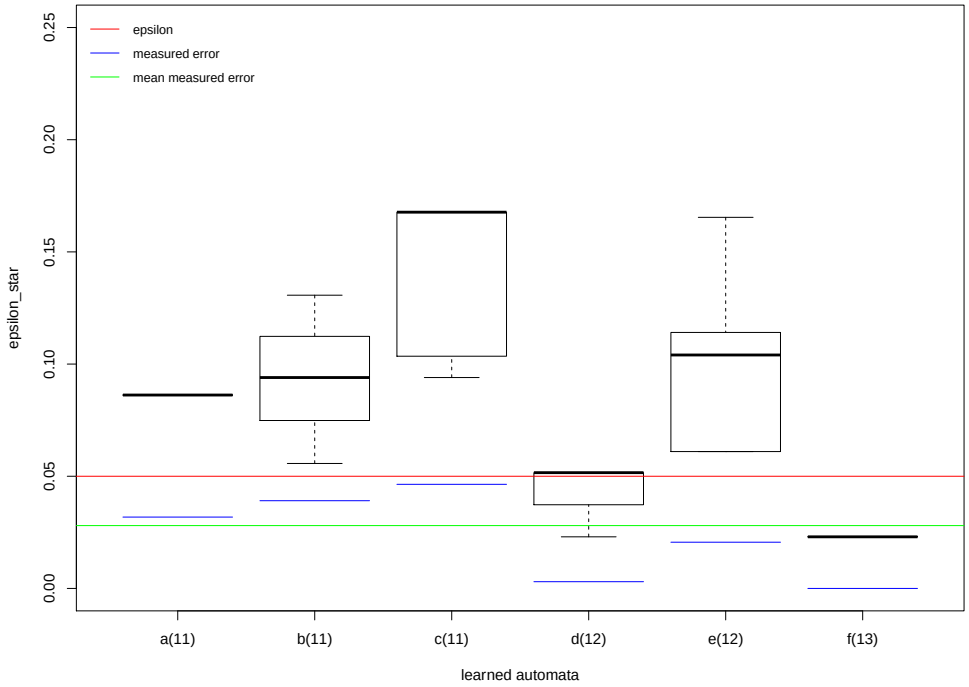


Figure 4.3: Measured error, ϵ, and ϵ_i^* for Example 4.2

by oracle **EQ** in each case. This is depicted in the figure with a boxplot of the ϵ_i^* values for each automaton produced. It is important to notice that the percentage of sequences in the symmetric difference (measured error) between each learned automaton and the neural network, measured on a set of randomly generated sample test sets, is always below the calculated ϵ_i^* value. It means that the measured empirical errors are consistent with the theoretical values. It is interesting to observe that the measured error was smaller than ϵ.

4.3 Example 3

This example presents the results obtained with an LSTM trained with a different dataset with positive and negative sequences of the same automaton as the previous example. In this case, the measured error was 0.3346.

We ran Bounded-L^* 20 times with $\epsilon = \delta = 0.05$, and a maximum query length of 12. All runs reached the bound. Fig. 4.5 summarizes the results obtained. The

experiment produced 11 different DFA, named with letters from a to k, with sizes between 4 and 27 number of states (indicated in parenthesis). Fig. 4.4 shows the 12-state automaton with smallest error. All automata exhibited measured errors smaller or equal than $\epsilon_i^\star$. In contrast, they were all above the proposed ϵ. For automata h (12 states) and j (24 states), measured errors are slightly smaller than the respective minimum ϵ_i^* values.

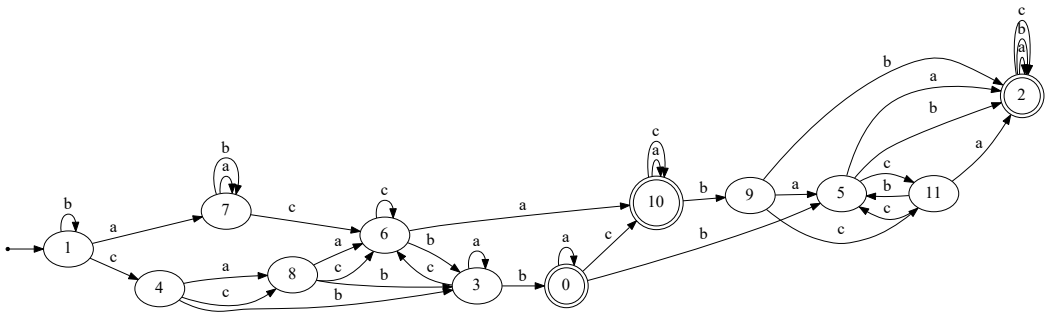


Figure 4.4: 12-state automaton with smallest error

Notice that this experiment shows higher variance in the number of automata, number of automaton states, and $\epsilon^\star$ values than the previous one.

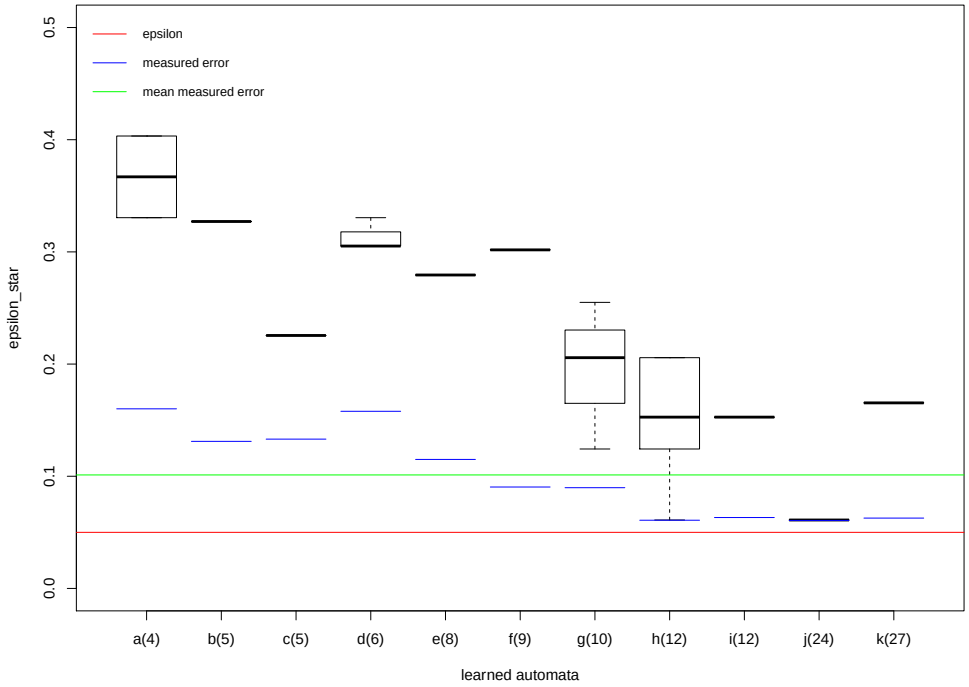


Figure 4.5: Measured error, ϵ, and ϵ_i^* of Example 4.3

4.4 Example 4

We study here the Alternating Bit Protocol (Fig. 4.6). The LSTM measured error was 0.2473. We ran Bounded-L^* with $\epsilon = \delta = 0.05$, and a maximum query length of 5.

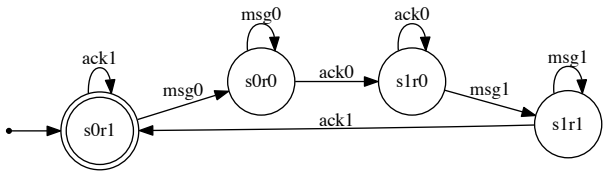


Figure 4.6: Alternating bit protocol automaton

Two different automata were obtained, named a and b, with 2 and 4 states

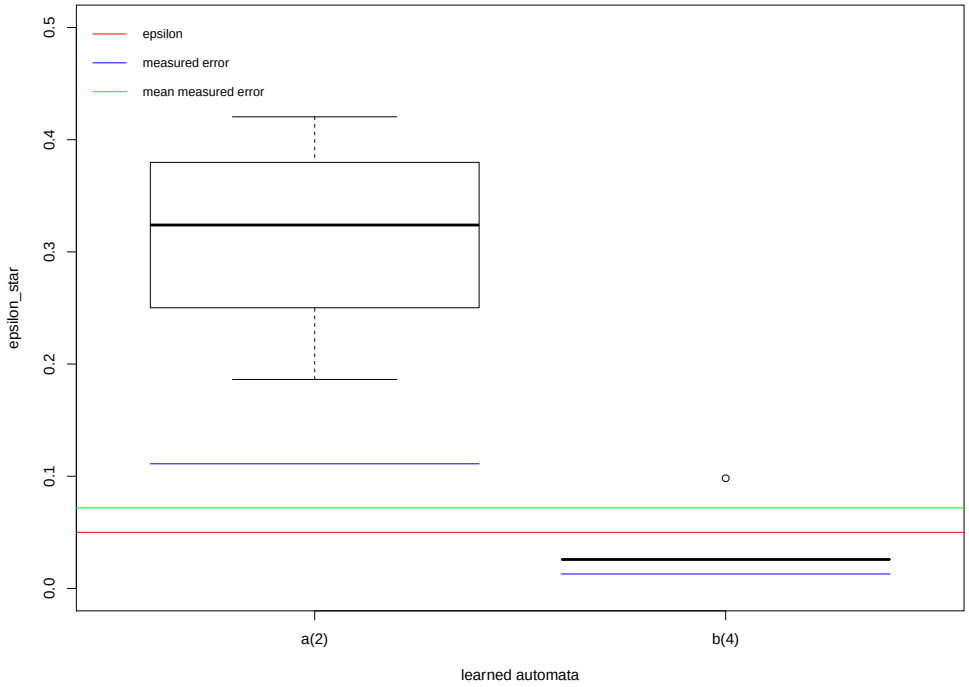
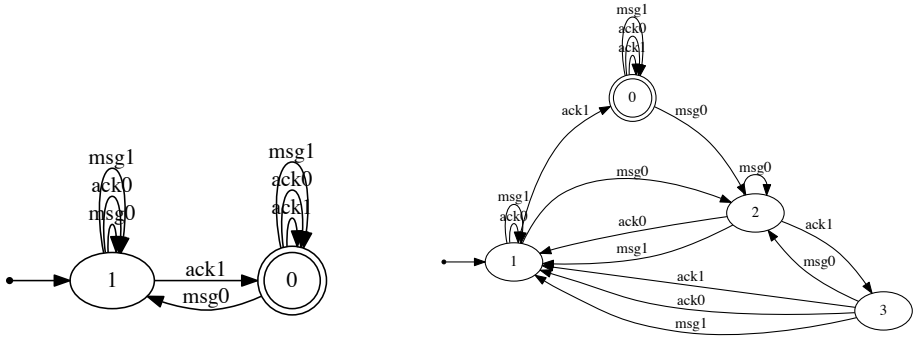


Figure 4.7: Measured error, ϵ, and ϵ_i^* of Exa. 4.4

respectively (Fig. 4.8). All runs that produced a reached the bound. This is explained in Fig. 4.7 where the boxplot for a is above ϵ. On the other hand, almost all runs that produced b completed without reaching the bound. Nevertheless, in all cases where the bound was reached, the sample size was big enough to guarantee $\epsilon^\star$ to be smaller than ϵ. Overall, the empirical error measures were consistent with the theoretical values.



(a) 2-state automaton (b) 4-state automaton

Figure 4.8: Generated automata for ABP

4.5 Example 5

We consider here an adaptation of the e-commerce website presented in [45] (Fig. 4.9). The labels are explained in the following table:

os: open session	ds: destroy session
gAP: get available product	eSC: empty shopping cart
gSC: get shopping cart	aPSC: add product to shopping cart
bPSC: buy products in shopping cart	

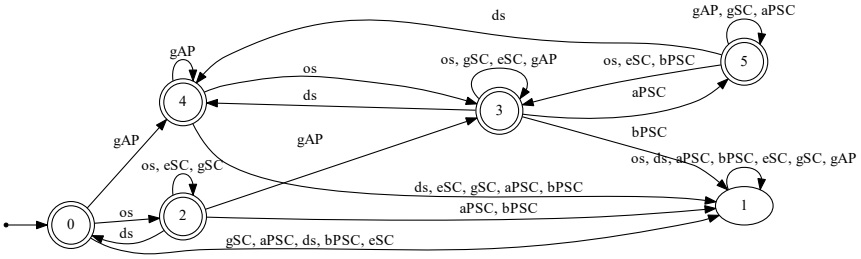


Figure 4.9: Model of the e-commerce example adaptation

The training set contained 44K sequences up to a maximum length of 16. The

test set contained 16K sequences. The measured error of the RNN on the test set was 0.0000625. We ran Bounded-L^* with $\epsilon = \delta = 0.05$, a maximum query length of 16 and a bound of 10 on the number of states. Fig. 4.10 shows the experimental results.

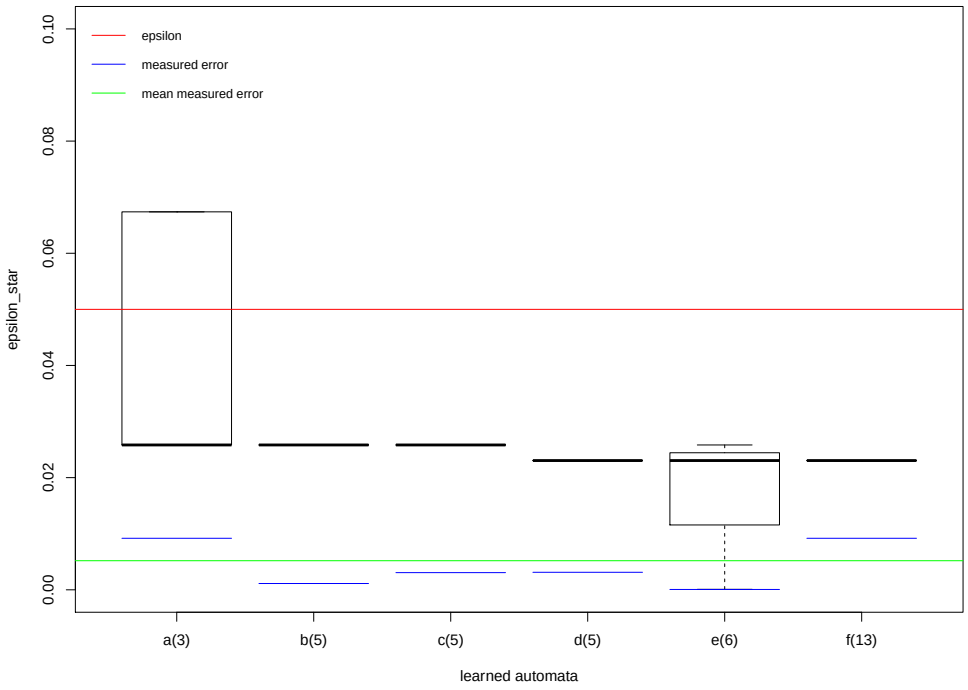


Figure 4.10: Measured error, ϵ, and ϵ_i^* of Example 4.5

Fig. 4.11 shows one of the learned automata. It helps interpreting the behavior of the network. For example, given $oS, gAP, aPSC, aPSC, bPSC$, the output of the network is 1, meaning that the sequence is a valid sequence given the concept the network was trained to learn. Besides, this sequence is also accepted by the automaton, yielding a traceable way of interpreting the result of the network. Moreover, for the input sequence $oS, gAP, aPSC, gSC, eSC, gAP, gAP, bPSC$, we find that the network outputs 1. However, this sequence is not accepted by the automaton (like the majority of the learned models). This highlights that the network misses an important property of the e-commerce site workflow: it is not possible to buy products when the shopping cart is empty.

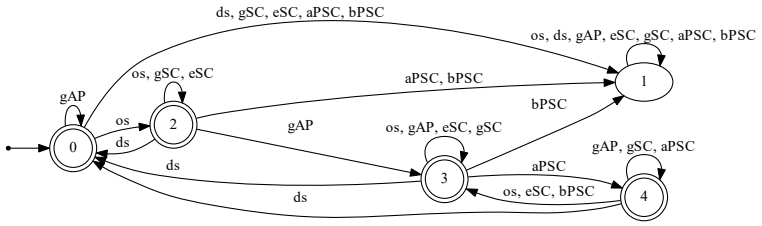


Figure 4.11: One of the learned models of the e-commerce example adaptation

Remark

The variance on the outcomes produced by Bounded-L^* for the same input RNN could be explained by representational, statistical and computational issues [46]. The first occurs because the language of the network may not be in the hypothesis space, due to the fact that RNN are strictly more expressive than DFA [42, 43]. The second and third are consequences of the sampling performed by **EQ** and the policy used to choose the counter-example.

5 Related Work

In this chapter the works that are, to the best of our knowledge, the closest related to ours are presented.

5.1 Rule Extraction from RNN

Rule extraction is defined by Craven [47] as 'Given a trained neural network and the data on which it was trained, produce a description of the network's hypothesis that is comprehensible yet closely approximates the network's predictive behavior'. Therefore, regular inference on RNN can be considered a particular case of rule extraction (RE) techniques, where the rules that are extracted are represented by a DFA. In this work we consider a more restrictive definition of the RE task, where there is no information available of the training data set.

Jacobsson [48] develops a survey on RE, following the *ADT taxomony* introduced by Andrews, Diederich and Tickle [49]. This taxonomy consists in five evaluation criteria which are:

1. Expressive power; which refers to the type of rules extracted.
2. Translucency; defined as 'degree to which the rule-extraction algorithm "looks inside" the ANN'.
3. Portability; meaning the extent to which the extraction algorithm is independent of the architecture of the network under analysis.
4. Rule quality; which refers to the performance for given metrics.
5. Algorithm complexity; referring to the time it takes to run an algorithm with relation to the input size.

Craven [47] also presents a taxonomy which can be considered a subset of the ADT one. Hence, we will continue to present related work using the latter.

With regard to expressive power, many works presenting different hypothesis spaces, that is types of rules have been developed. For instance Craven [47] focuses in extracting decision trees and sparse perceptrons. The approaches discussed by Bologna et al in [50] are devoted to extracting decision trees and 'if-else' rules for specific classes of multi-layer feed-forward ANN. Such models are less expressive than DFA when it comes to modelling sequences. Work that extract DFAs are discussed in the next section.

If we dig into the translucency criteria, four categories are distinguished:

1. Compositional approaches, where rules are constructed based on the hidden layers of the RNNs.
2. Decompositional approaches, where rules are constructed based on individual neurons.
3. Pedagogical approaches, which construct rules by regarding the target RNN as a black box and have no access to the inner state of this RNN
4. Eclectic approaches, which represent a hybrid of decompositional and pedagogical approaches.

In the decompositional approach area, some works present techniques where individual rules are extracted from data instances [51, 52, 53]. However, this kind of technique is not related to ours, as it correspondes to another case of explainability. There is no mention of any eclectic approach algorithm in [48].

Herein after we focus on compositional and pedagogical approaches that extract DFAs from RNNs.

5.2 DFA Extraction

Works that use the compositional approach are white box, that is, they rely on knowing the internal structure of the RNN. A general description of the compositional rule extraction from RNN techniques can be described as:

1. Quantisation of the continuous state space of the RNN, resulting in a discrete set of states.

2. State and output generation (and output classification, if necessary) by feeding the RNN input patterns. Output generation does not apply to DFAs.

3. Rule construction based on the observed state transitions. These rules are the transition function, in the DFA scenario.

4. Rule set minimization. When extracting DFAs, it is DFA minimization.

Compositional approaches have been studied by Giles and coauthors [26, 54]. For instance, Giles et al [26] focus on extracting rules from second-order RNN. This work has been recently extended by Wang et al. in [54], where they continue with the work but center the evaluation in the stability of the rule extraction method using Tomita grammars [55] as frame of reference. Wang et al. in [56] do a comparative study regarding extraction of DFA from not only second-order RNN but also other architectures also trained on Tomita grammars. In this line of ideas, the algorithm developed in [24] proposes an equivalence query based on the comparison of the proposed hypotheses with an abstract representation of the RNN that is obtained through an exploration of its internal state.

Regarding pedagogical approaches, the only work that strictly falls in this category is the one by Vahed and Omlin [57]. It is considered black box as it does not rely on the internal state of the RNN, however the data used for extraction was based on all strings up to a given length and fed to the 'Trakhtenbrot-Barzdin' algorithm [58].

When it comes to portability, the analysis remains similar to the one presented for translucency. This is because depending on internal architecture characteristics limits the ability of transfering the techniques to other types of RNN.

Among all metrics for rule quality mentioned in the literature, the most appropriate in our context is 'fidelity', defined as the accuracy of the extracted DFA with respect to the RNN veredict, that is $\mathbb{P}\left(\mathcal{L}(\mathcal{N}) \cap \mathcal{L}(\mathcal{A})\right)$ which is equal to $1 - \mathbb{P}\left(\mathcal{L}(\mathcal{N}) \oplus \mathcal{L}(\mathcal{A})\right)$.

Vahed and Omlin in [57] present empirical evidence that black box approaches (pedagogical) could lead to better fidelity than clustering based ones (compositional). However their algorithm fits into Craven definition of RE, that is, it relies not only on the network output but also on the data set used for training it.

A thorough analysis of rule quality in this field is still to be developed. Recently Wang et al [54, 56] have started looking at this question restricted to compositional approaches.

In contrast to all the aforementioned papers, our work is the first one to propose a mathematical analysis of rule extraction fidelity. By resorting to PAC we are able to provide upper bounds on the probability of divergence, meaning lower bounds on the fidelity of the output DFA.

Finally, taking as an example the work by Vahed and Omlin in [57] Jacobson [48] argues that algorithm complexity could be a drawback of black box approaches. This is because despite the learning algorithm itself is of polynomial time complexity on the size of the prefix tree, this size is $O(|\Sigma|^l)$, where l is the depth of the tree. As a consequence, their approach is not likely to scale up to problems with large sets of symbols. Large alphabets may also be an issue for L^*algorithm, however it can be mitigated resorting to Maler and Mens work [59], which present an extension of the L^* algorithm for learning languages over alphabets of large size. On the other hand, the complexity of cluster analysis (typically *k-means*) present in white box approaches [26, 54, 24] is theoretically *NP-hard* [60], but polynomial in practice by the use of heuristics whose downside is that they may have negative impact in terms of fidelity [57].

The following table (5.1) summarizes the closest works to ours, we will consider only the works that extract DFAs in order to make the comparison succint, therefore the row 'expressive power' is ommited.

	Bounded L* [1]	**Giles et al [26, 54]**	**Omlin & Vahen [57]**	**Weiss et al [24]**
T	Pedagogical	Compositional	Pedagogical	Compositional
P	No restrictions	Restrictions on architecture	No restrictions	Restrictions on architecture
RQ	Presents PAC guarantees	Open question	Open question	Open question
AC	Polynomial on maximum number of states and maximum length of any counterexample	Theoretically *NP-hard* but polynomial in practice due to the use of heuristics	Polynomial on depth of prefix tree, exponential on size of Σ	Theoretically *NP-hard* but polynomial in practice due to the use of heuristics

Table 5.1: Comparison of related work according to *ADT taxonomy*. **T** for 'translucency', **P** for 'portability', **RQ** for 'rule quality' and **AC** for 'algorithm complexity'.

5.3 Other related Work

Other interesting and recent works in the field are: Rabusseau et al [61] recently proved the relationship one to one between second order recurrent neural networks and weighted automata, this result has deep implications in the expressivity and means of training of such models.

Work on regular inference [44] focused on studying the learnability of different classes of automata but none was applied to extracting them from RNN.

As a conclusion of this chapter we can say that none of the related works provide means for active black-box model explanation in the context of RNN. On the contrary, our approach using PAC regular inference is the first approach that is completely agnostic of the underlying model of the black box and at the same time studies it in an active learning setting.

6 Conclusions

We presented an active PAC-learning algorithm for learning automata that are approximately correct with respect to neural networks. Our algorithm is a variant of Angluin's L^* where a bound on the number of states or the query length is set to guarantee termination in application domains where the language to be learned may not be a regular one. We also studied the error and confidence of the hypotheses obtained when the algorithm stops by reaching a complexity bound with k greater than zero divergencies between the network and the generated automaton.

The experimental evaluation of our implementation showed that the approach is able to infer automata that are reasonable approximations of the target models with high confidence, even if the output model does not pass the usual 0-divergence **EQ** statistical test of the PAC framework. These evaluations also provided empirical evidence that the method exhibits high variability in the generated DFAs. This is a key concern to be addressed in future work.

The variance on the outcomes produced by Bounded-L^* is still an open research line. A preliminary work by Diego Zuluaga and Kevin Chacón addressed this issue with no positive results so far. In [62] they proposed a series of different criteria to select the counterexample returned by the **EQ**. However there was no empirical evidence showing that selection to be better than the shortest-one criterion used in this work.

Acknowledgments

This work has been partially funded by an ICT4V - Information and Communication Technologies for Verticals master thesis grant under the code `POS_ICT4V_2016_1_06`.

7 Bibliography

[1] F. Mayr and S. Yovine, "Regular inference on artificial neuralnetworks," in *Machine Learning and Knowledge Extraction*, A. Holzinger *et al.*, Eds. Cham: Springer International Publishing, 2018, pp. 350–369.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org.

[3] A. Carrington, P. Fieguth, and H. Chen, "Measures of model interpretability for model selection," in *Machine Learning and Knowledge Extraction*, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Cham: Springer International Publishing, 2018, pp. 329–349.

[4] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, p. 211–222, 05 2017.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436 – 444, 2015.

[6] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing neural predictions," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[7] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable ai systems for the medical domain?" *arXiv:1712.09923*, 2017.

[8] T. G. Calderon and J. J. Cheh, "A roadmap for future neural networks research in auditing and risk assessment," *International Journal of Accounting Information Systems*, vol. 3, no. 4, pp. 203 – 236, 2002, second International Research Symposium on Accounting Information Systems.

[9] C. Zhang, J. Jiang, and M. Kamel, "Intrusion detection using hierarchical neural networks," *Pattern Recognition Letters*, vol. 26, no. 6, pp. 779 – 791, 2005.

[10] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An approach to evaluating interpretability of machine learning," *CoRR*, vol. abs/1806.00069, 2018.

[11] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti, "A survey of methods for explaining black box models," *CoRR*, vol. abs/1802.01933, 2018.

[12] A. Holzinger, M. Plass, K. Holzinger, G. C. Crisan, C.-M. Pintea, and V. Palade, "A glass-box interactive machine learning approach for solving np-hard problems with the human-in-the-loop," *arXiv:1708.01104*, 2017.

[13] D. Gunning, "Darpa's explainable artificial intelligence (xai) program," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19. New York, NY, USA: ACM, 2019, pp. ii–ii. [Online]. Available: http://doi.acm.org/10.1145/3301275.3308446

[14] C. Zhou, B. Cule, and B. Goethals, "Pattern based sequence classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1285–1298, May 2016.

[15] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 40–48, Nov. 2010.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[17] F. A. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, Nov 2001.

[18] L. Deng and J. Chen, "Sequence classification using the high-level features extracted from deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6844–6848.

[19] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," *CoRR*, vol. abs/1511.08630, 2015.

[20] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia*, April 19-24, 2015, pp. 1916–1920.

[21] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings of 23rd European Symp. on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[22] Y. Wang and F. Tian, "Recurrent residual learning for sequence classification," in *Proc. 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA*, November 1-4, 2016, pp. 938–943.

[23] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784, April 2014.

[24] G. Weiss, Y. Goldberg, and E. Yahav, "Extracting automata from recurrent neural networks using queries and counterexamples," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm, Sweden: PMLR, 10–15 Jul 2018.

[25] C. W. Omlin and C. L. Giles, "Constructing deterministic finite-state automata in recurrent neural networks," *J. ACM*, vol. 43, no. 6, pp. 937–972, Nov. 1996.

[26] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, and Y. C. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Comput.*, vol. 4, no. 3, pp. 393–405, May 1992.

[27] K. Gorman and R. Sproat, "Minimally supervised number normalization," *TACL*, vol. 4, pp. 507–519, 2016.

[28] M. Grzes and M. Taylor, Eds., *Proceedings of the Adaptive and Learning Agents Workshop*, 2010.

[29] C. de la Higuera, *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[30] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, Nov. 1984.

[31] D. Angluin, "Computational learning theory: Survey and selected bibliography," in *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '92. New York, NY, USA: ACM, 1992, pp. 351–369.

[32] K. Murphy, "Passively learning finite automata," Santa Fe Institute, Tech. Rep. 96-04-017, 1996.

[33] E. M. Gold, "Complexity of automaton identification from given data," *Information and Control*, vol. 37, no. 3, pp. 302 – 320, 1978.

[34] D. Angluin, "Learning regular sets from queries and counterexamples," *Inf. Comput.*, vol. 75, no. 2, pp. 87–106, Nov. 1987.

[35] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[36] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, Sep. 1956.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699. [Online]. Available: http://dl.acm.org/citation.cfm?id=65669.104451

[38] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 03 1990.

[39] J. F. Kolen and S. C. Kremer, *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*. IEEE, 2001. [Online]. Available: https://ieeexplore.ieee.org/document/5264952

[40] R. S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, pp. 111–161, 02 1983.

[41] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.

[42] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 440–449. [Online]. Available: http://doi.acm.org/10.1145/130385.130432

[43] M. Suzgun, Y. Belinkov, and S. M. Shieber, "On evaluating the generalization of lstm models in formal languages," *CoRR*, vol. abs/1811.01001, 2018.

[44] J. Heinz, C. de la Higuera, and M. van Zaanen, "Formal and empirical grammatical inference," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 2:1–2:83.

[45] M. Merten, "Active automata learning for real life applications," Ph.D. dissertation, Technischen Universität Dortmund, 2013.

[46] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, ser. MCS '00. London, UK, UK: Springer-Verlag, 2000, pp. 1–15.

[47] M. W. Craven, "Extracting comprehensible models from trained neural networks," Ph.D. dissertation, The University of Wisconsin - Madison, 1996, aAI9700774.

[48] H. Jacobsson, "Rule extraction from recurrent neural networks: A taxonomy and review," *Neural Computation*, vol. 17, pp. 1223–1263, 06 2005.

[49] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowl.-Based Syst.*, vol. 8, pp. 373–389, 1995.

[50] G. Bologna and Y. Hayashi, "Characterization of symbolic rules embedded in deep dimlp networks: A challenge to transparency of deep learning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 7, no. 4, pp. 265–286, 2017.

[51] W. J. Murdoch and A. Szlam, "Automatic rule extraction from long short term memory networks," *CoRR*, vol. abs/1702.02540, 2017.

[52] W. J. Murdoch, P. J. Liu, and B. Yu, "Beyond word importance: Contextual decomposition to extract interactions from lstms," *CoRR*, vol. abs/1801.05453, 2018.

[53] C. Singh, W. J. Murdoch, and B. Yu, "Hierarchical interpretations for neural network predictions," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SkEqro0ctQ

[54] Q. Wang, K. Zhang, A. G. Ororbia, II, X. Xing, X. Liu, and C. L. Giles, "An empirical evaluation of rule extraction from recurrent neural networks," *Neural Comput.*, vol. 30, no. 9, pp. 2568–2591, Sep. 2018. [Online]. Available: https://doi.org/10.1162/neco_a_01111

[55] M. Tomita, "Dynamic construction of finite automata from examples using hill-climbing," in *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, Ann Arbor, Michigan, 1982, pp. 105–108.

[56] Q. Wang, K. Zhang, A. G. O. II, X. Xing, X. Liu, and C. L. Giles, "A comparison of rule extraction for different recurrent neural network models and grammatical complexity," *CoRR*, vol. abs/1801.05420, 2018. [Online]. Available: http://arxiv.org/abs/1801.05420

[57] A. Vahed and C. W. Omlin, "A machine learning method for extracting symbolic knowledge from recurrent neural networks," *Neural Computation*, vol. 16, no. 1, pp. 59–71, 2004. [Online]. Available: https://doi.org/10.1162/08997660460733994

[58] B. A. Trakhtenbrot and I. M. Barzdin, *Finite automata : behavior and synthesis [by] B. A. Trakhtenbrot and Ya. M. Barzdin. Translated from the Russian by D. Louvish. English translation edited by E. Shamir and L. H. Landweber.* North-Holland Pub. Co.; American Elsevier Amsterdam, New York, 1973.

[59] O. Maler and I.-E. Mens, "Learning regular languages over large alphabets," in *Tools and Algorithms for the Construction and Analysis of Systems*, E. Ábrahám and K. Havelund, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 485–499.

[60] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *Proceedings of the 3rd International Workshop on Algorithms and Computation*, ser. WALCOM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 274–285. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00202-1_24

[61] G. Rabusseau, T. Li, and D. Precup, "Connecting weighted automata and recurrent neural networks through spectral learning," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 1630–1639. [Online]. Available: http://proceedings.mlr.press/v89/rabusseau19a.html

[62] K. Chacón and D. Zuluaga, "Búsqueda inteligente de contraejemplos para la inferencia de lenguajes," Universidad ORT, Uruguay, Tech. Rep., 2018. [Online]. Available: https://bibliotecas.ort.edu.uy/bibid/88071