How Chaotic Are Recurrent Neural Networks?

Pourya Vakilipourtakalou, Lili Mou
Department of Computing Science, University of Alberta
Alberta Machine Intelligence Institute (Amii)
vakilipo@ualberta.ca
doublepower.mou@gmail.com

Abstract

Recurrent neural networks (RNNs) are non-linear dynamic systems. Previous work believes that RNN may suffer from the phenomenon of *chaos*, where the system is sensitive to initial states and unpredictable in the long run. In this paper, however, we perform a systematic empirical analysis, showing that a vanilla or long short term memory (LSTM) RNN does not exhibit chaotic behavior along the training process in real applications such as text generation. Our findings suggest that future work in this direction should address the other side of non-linear dynamics for RNN.

1 Introduction

Recurrent neural networks (RNNs), e.g., vanilla RNN and Long Short-Term Memory (LSTM, Hochreiter & Schmidhuber, 1997), are important architectures for sequential data processing. In natural language processing (NLP), for example, an RNN is used to not only learn the representation of text but also generate a sentence in a certain task (Ilya Sutskever, 2014).

An RNN works in the following way: the RNN keeps a fixed dimensional hidden state. At each time step, it reads in an input signal, updates its state accordingly, and makes a prediction if needed. We are particularly interested in RNNs applied to text generation, where at each step, the previously generated word is fed as input and the RNN predicts the next word.

Mathematically speaking, such RNNs are actually an *iterative map*, or a *map* (Strogatz, 2018), from the space of the hidden state to itself. This means that the hidden state is iteratively updated at every step (based on the previous hidden state) by the RNN transition.

It is noted that, for a typical RNN like vanilla transition and LSTM, the map is *non-linear*, and, a potential known problem of non-linear iterative maps is its *chaotic behavior* (T. Alligood et al., 2000). Usually, a chaotic map is not periodic in the limit of the map being iterated for infinitely many times. Also, a chaotic map prevents an accurate prediction of its state for future steps, in which case we say that the map is attracted to an *strange attractor*.

Analyzing the chaotic behavior of an RNN becomes a fundamental scientific question for its applications, as chaos behavior might affect how stable an RNN is or how well the RNN performs. There have been a few studies on the chaotic behavior of RNNs.

Both vanilla and LSTM RNNs are shown to be chaotic with certain parameters (Bertschinger & Natschlager, 2004; Laurent & von Brecht, 2016), which are usually assigned by humans in a low dimensional (e.g., 2D) hidden state. Based on such results, a few regularization methods are proposed to make RNNs less chaotic (Laurent & von Brecht, 2016; Chang et al., 2019). For simplicity, such studies also ignore the input to the RNN at every step.

In this paper, we analyze the chaos of RNNs (vanilla transition and LSTM) in a more realistic setting in a practical text generation task. We show empirically that—along the entire training process from random weights to a well trained model—the RNN does not exhibit chaotic behavior. Moreover, we experimented the settings of with and without input; empirical results show that, if an RNN is not fed with input, its hidden state is almost always attracted to a single fixed point.

Our unexpected results also imply that, for future work, we should adopt a realistic setting to understand the typical behavior of an RNN in real-world applications.

2 Formulation

An *iterative map*, or a *map*, is a function $f : S \to S$ from one space to the same space. An RNN can be formulated as $\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$, where $\boldsymbol{h}_t$ is the hidden state for the tth step ($\boldsymbol{h}_0$ being the initial hidden state) and $\boldsymbol{x}_t$ is the input. In text generation, for example, we feed in the previously generated word as input, which is usually the most probable word, i.e., $x_t = \arg\max p(x_{t-1}|\boldsymbol{h}_{t-1})$. This means that such RNN is an iterative map because $\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{h}_{t-1}, \arg\max p(\cdot|\boldsymbol{h}_{t-1}))$. In previous work where researchers ignore input of RNN, we have $\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{0})$, indicating that in this setting the RNN is also an iterative map. These are two scenarios we would explore in this paper, and we refer to them as "*with input*" and "*without input*," respectively.[1]

An important notion in the study of iterative maps is the *orbit*, which is the sequence of function values of a map, given an initial input. In the context of an RNN with a given initial hidden state $\boldsymbol{h}_0$, the orbit is $(\boldsymbol{h}_0, \boldsymbol{h}_1, \boldsymbol{h}_2, \cdots)$.

An orbit may exhibit different behaviors. If $f^k(p) = p$, where f^k means $f \circ \cdots \circ f$ for k times, we say the orbit containing p is *periodic* with a period of k, or a *period-k orbit*. Especially, if $k = 1$, the point p is called a *sink*. If an orbit $(\boldsymbol{h}_0, \boldsymbol{h}_1, ..., \boldsymbol{h}_t)$ gets closer to a periodic orbit as $t \to \infty$, we say the orbit is *asymptotically periodic*.

An orbit may also be *chaotic*. While different researchers do not agree with a common definition (Strogatz, 2018; T. Alligood et al., 2000), a chaotic orbit usually means that the orbit $(\boldsymbol{h}_0, \boldsymbol{h}_1, \cdots)$ is sensitive to $\boldsymbol{h}_0$, and $(\boldsymbol{h}_0, \boldsymbol{h}_1, \cdots)$ must not be asymptotically period.

In text generation in our experiments, the hidden state of an RNN is usually a high-dimensional space, and it may be difficult to detect periodicity based on RNN's hidden state. We instead approximate it by the RNN's output words. That is, $p(y_t|\boldsymbol{h}_t) = \mathrm{softmax}(W_o\boldsymbol{h}_t + b_o)$, where W_o and b_o are the output parameters. We choose the most probable word as the predicted word $y_t = \arg\max p(y_t|\boldsymbol{h}_t)$. In this case, we call $(y_1, y_2, \cdots)$ an orbit in the output space.

It is noted that the orbit $(y_1, y_2, \cdots)$ being periodic is a necessary condition of $(\boldsymbol{h}_0, \boldsymbol{h}_1, \cdots)$ being periodic. However, we will show that such approximation would be fairly accurate by plotting the latent space with dimensionality reduction. Thus, we will compute the period (if existing) of an RNN map by the period of output words in our experiments.

For example, if an RNN generates a sequence "*I like it but , I like it but ,* $\cdots$" for a fairly large number of steps (e.g., 2K–20K), we would empirically estimate the period of this orbit as 5, and we will show that the hidden states are indeed trapped in 5 points.

3 Experiments

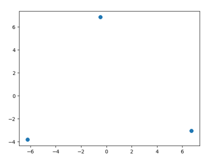
Setup. We use the Anna Kerenina textbook as our training corpus.[2] The corpus has 400K words, and we split the dataset by 4:1 for training and validation (mainly for early stop). The test phase is to generate sentences following the trained language model, which does not involve input.

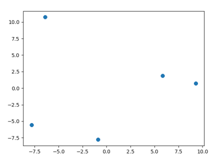
To analyze different types of RNNs, we experimented with the vanilla and the LSTM transitions. The size of the hidden state and embedding size, in both cases, were 300 and 500, respectively. We used Adam as the training algorithm with the initial learning rate 0.001 and other default hyperparameters.

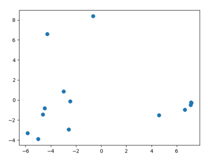
Our analysis of how chaotic an RNN is involves running the RNN for a large number of times with different initial hidden states $\boldsymbol{h}_0$. The treatment was slightly different. For the RNN without input, we randomly sampled $\boldsymbol{h}_0$ from a Gaussian distribution centered at $\boldsymbol{0}$. For RNN with input, we experimented with different initial words x_1 fed as input for the first step. In this case, the RNN

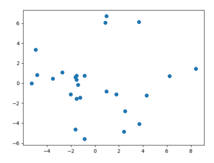
[1] In other usages of RNN (for example, sentence encoding), the input x_t is a word in a given sentence. Such RNN is not well defined as an iterative map, because the function from $\boldsymbol{h}_{t-1}$ to $\boldsymbol{h}_t$ is subject to x_t.

[2] `https://www.kaggle.com/wanderdust/anna-karenina-book`

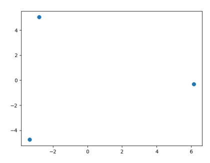

period-3 orbit

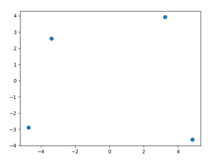

period-5 orbit

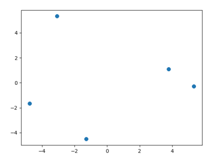

period-14 orbit

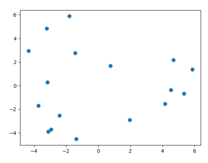

period-26 orbit

(a) Orbits of vanilla RNN


period-3 orbit


period-4 orbit


period-5 orbit


period-17 orbit

(b) Orbits of LSTM

Figure 1: Hidden state of a) vanilla RNN and b) LSTM with input traps in a periodic orbit.

		Epoch 0	Epoch 10	Epoch 20	Epoch 30	Epoch 40
Vanilla RNN	Average Period	4	39	234	182	101
	Non-Periodic	0%	0%	0%	0%	0%
LSTM	Average Period	3	27	2408	3713	4018
	Non-Periodic	0%	0%	0%	0%	0%

Table 1: Vanilla RNN and LSTM trained with input are used to generate words, using the entire words in the vocabulary as input. This table lists the average period of the periodic orbits that the hidden states trap into and also the percentage of the initial words of the vocabulary that perform non-periodic behavior.

could be thought of as an iterative map, starting from the second step, with different initial hidden states $\boldsymbol{h}_1$. In total, we had 15K trajectories (different runnings of RNN) in our experiments for each setting.

We detect the period of RNNs by its output words. We ran each trajectory by 2K–20K steps, and then verified the period by another 2K–40K steps. For the setting with longest periods (Table 1), a periodic orbit is verified by roughly 10 times.

Results of RNNs with input. We first analyze the behavior of an RNN with input, which is the more realistic usage in applications. We train the model on the training set while monitoring the validation perplexity. We obtained 139 perplexity for vanilla RNN and 108 perplexity for LSTM. We compared our results to previous work (Mikolov et al., 2010), and although the corpora are different, we obtained a reasonable perplexity of English.

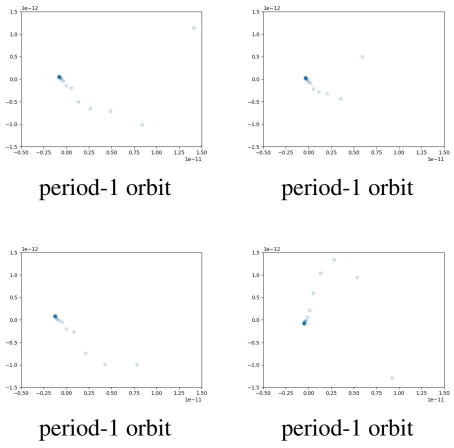
Unexpectedly, we observe that a well-trained RNN (with input) does not exhibit any chaotic behavior. This is shown in Figure 1, where we randomly select a few trajectories and plot the high-dimensional hidden state into a 2D space with principle component analysis (PCA, Bishop, 2006). In each plot, we have more than 2K points, but in fact, they are periodic with a length of 3–26.

We are curious if the training process would affect the periodic or chaotic behavior of an RNN. Thus, we quantitatively report with different words the average period of an RNN and the percentage of non-periodic orbits in Table 1. It is seen that all the orbits we obtained are attracted by periodic orbits, showing that they are not chaotic.

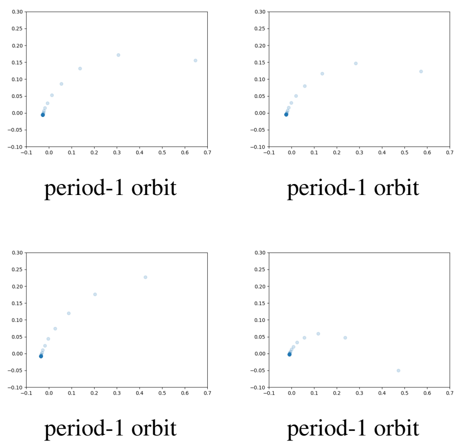
It is also seen that, for both vanilla RNN and LSTM, the average period at the beginning of training is short and then the period becomes longer up to Epoch 20. For vanilla RNN, the average period decreases if we train it more, but we do not observe such decrease for LSTM. Also, the average period of LSTM is significantly longer than that of a vanilla RNN after 20 epochs.

Period	Vanilla RNN	LSTM
1	100%	100%
> 1	0%	0%
non-periodic	0%	0%

Table 2: Vanilla RNN and LSTM trained without input are used to generate words, with different initialization of the hidden state sampled from a normal distribution between 0 and 1. This table lists the percentage of the initial hidden states that resulted in the specified periodic outputs.



(a) Hidden state of vanilla RNN



(b) Hidden state of LSTM

Figure 2: Hidden state of a) vanilla RNN and b) LSTM without input is attracted to a sink.

Results of RNNs without input. We also analyzed the behavior of an RNN without input. This is the typical setting of previous work on analyzing RNN's chaotic behavior (Bertschinger & Natschlager, 2004; Laurent & von Brecht, 2016).

We show the period of such RNNs in Table 2. Again, we do not observe any chaotic behavior for both vanilla RNN and LSTM. More surprisingly, all these orbits are attracted by a sink, i.e., the period is 1. This is further confirmed by the PCA plot of RNN's hidden states (Figure 2). As shown, all the hidden states are monotonically approaching a single point, which is a sink of the RNN.

In comparison to RNN with input, we see that the period of RNN without input is much shorter, and in fact, all orbits are period-1. Our conjecture is that, if a word is fed to RNN as an input, it is taken from the argmax of the predicted probability in the previous step. Such input word is a discrete token chosen from the vocabulary. This, in turn, may drag the RNN's hidden state and make the period longer.

It is also noted that a few previous papers report the chaotic behavior of RNN, which seemingly contradicts the observation of our paper. We contacted by personal email, and with the help of them, we replicated the chaos with certain RNN weights. Considering the evidence in previous work and our paper, we conclude that vanilla RNN or LSTM could be chaotic with certain weights, but in a realistic setting where the RNN weights are either randomly initialized or well trained, the RNN does not exhibit chaotic behavior.

4 Conclusion and Future Work

In this paper, we show that an RNN with either vanilla or LSTM transition is *not* chaotic along the training process in real applications. Our results contradict previous belief that RNNs are chaotic non-linear dynamic systems, although previous work indeed shows that RNNs could be chaotic with certain magic weights.

Our findings also suggest that, in future work, it is unnecessary to encourage non-chaos for an RNN, because it does not exhibit chaos in real applications. On the contrary, we observe that a better RNN typically exhibits a longer period (e.g., LSTM vs. vanilla RNN, well-trained vs. randomly initialized). We conjecture that, in real applications of RNNs, chaos should be encouraged, rather than discouraged, so that the RNN is not attracted to a periodic orbit.

Acknowledgments

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No. RGPIN-2020-04465. Lili Mou is also supported by AltaML, the Amii Fellow Program, and the Canadian CIFAR AI Chair Program.

References

Nils Bertschinger and Thomas Natschlager. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–36, 2004.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. In *ICLR*, 2019.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Oriol Vinyals Ilya Sutskever. Sequence to sequence learning with neural networks. In *ICLR*, 2014.

Thomas Laurent and James von Brecht. A recurrent neural network without chaos. 2016.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

Steven H. Strogatz. *Nonlinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, Boca Raton, 2018.

Kathleen T. Alligood, Tim D. Sauer, and James A. Yorke. *Chaos: An Introduction to Dynamical Systems*. Springer Science & Business Media, 2000.