Explicit Memorization for Recurrent Neural Networks with Autoencoders

Antonio Carta[1 [0000−0002−0003−2323]]

Università di Pisa, Pisa, Italia
antonio.carta@di.unipi.it

Abstract. Recurrent neural networks are difficult to train due to the vanishing and exploding gradient problem. Most of the solutions in the literature revolve around the design of new models able to mitigate this issue. However, they ignore the training algorithm, relying on gradient descent and end-to-end training. In this extended abstract, we propose a conceptual separation of recurrent models into two components: *a feature extractor and a memory*. We introduce the Linear Memory Network, a recurrent model based on this conceptual framework. The separation of the two components allows us to concentrate on the development of better memory models and training algorithms. We exploit this model to design several algorithms to train the LMN and its hierarchical extension and initialize the memory based on the optimal solution of the linear autoencoder for sequences. After the initialization, the autoencoder is implicitly used as an explicit memory by encoding and decoding the entire sequence of hidden states in its memory. The experimental results show that using these algorithms, designed for memorization, we can improve the results of recurrent models on a variety of tasks.

Keywords: Recurrent Neural Networks · Autencoders · modular neural networks.

1 Introduction

Recurrent neural networks (RNN)[6] solve sequential problems by iteratively updating an internal state at each timestep. Recently, RNNs obtained state-of-the-art results in several sequential domains, such as speech recognition [7] and machine translation [12]. Despite their success, RNNs are extremely difficult to train due to the vanishing gradient problem[10], which makes it extremely hard to learn long-term dependencies.

The work outlined in this extended abstract focuses on the development of a novel memorization mechanism for recurrent neural networks. The objectives of this work are:

- the design of *novel RNN models* with explicit memorization;
- the design of *novel training algorithms* for the RNN memory;
- the study of the tradeoffs between a pure memorization approach for sequential problems and end-to-end training.

CEUR-WS.org/Vol-2495/paper11.pdf

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Currently, recurrent architectures are trained end-to-end by stochastic gradient descent. Most of the work in the literature try to improve current recurrent models by proposing architectural changes. Instead, our proposal is based on two principles: the separation between memory and functional components, and the development of specialized training algorithms for recurrent networks, a field largely ignored by the current literature.

2 State of the Art

Several solutions have been proposed to mitigate the vanishing and exploding gradient problem. *Gated architectures*, like LSTM[11] and GRU[5], try to solve this problem modifying the model architecture to reduce the vanishing gradient (but without eliminating the problem). *Orthogonal models*[14, 17, 1] solve the problem by exploiting orthogonal linear transformations and linear activation functions, which guarantee the constant gradient propagation. However, orthogonal models still perform worse than gated architectures on several tasks.

Attention models[3] solve the problem by performing a weighted sum of the entire sequence of previous hidden states. This approach solves the vanishing gradient problem but it is much more computationally expensive and does not scale to long sequences. *Memory-Augmented Neural Networks (MANN)*[8, 9] are a class of models made of a controller and an external memory, where the controller can read and write the memory through an interface. These models try to solve the memorization problem on a model-definition level. While they obtained highly promising results on some synthetic tasks, they are extremely hard to train. Furthermore, since the model is trained end-to-end there are no guarantees that the resulting model will use the external memory in any meaningful way.

Linear autoencoders for sequences (LAES)[15] are a fundamental component of our proposed model. The optimal solution of a LAES can be easily found with a closed-form expression[15]. Furthermore, the literature provides some equivalence results which bridge the gap between feedforward networks, which can see the entire sequence at once, and recurrent neural networks[16].

3 Problem Statement

We separate the problem of processing sequential data into two subproblems:

functional problem the problem of extracting informative features from the current input, given the current state of the memory;

memorization problem the problem of encoding the features extracted by the functional component into a memory.

The memorization module is a recurrent component and therefore can suffer from the vanishing gradient problem. We focus our work on the development of new models and training algorithms for this component. Dividing sequential problems into two separate subproblems allows building separated solutions

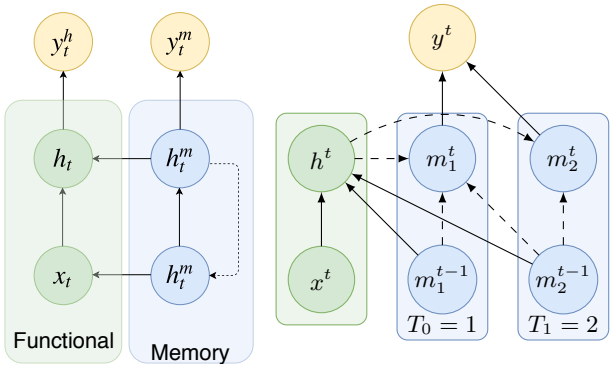

Fig. 1: LMN and MultiScale LMN architecture.

optimized for the peculiarities of the task, including novel architectures and training algorithms. Furthermore, each component can be easier since it must solve only part of the problem. As an example, the Linear Memory Network [2], one of our proposed models, is made of a feedforward network and a linear recurrent network, two components which are less complex than a monolithic RNN.

Another interesting property of the separation is that it allows us to concentrate on the memorization task. What are the limits of memorization? How can we learn when and what to forget? These questions become easier to address if we focus only on the memorization subtask.

4 Results

To investigate the separation of the memory from the model architecture we introduced the *Linear Memory Network* (LMN)[2], a novel recurrent model where the functional component is implemented using a feedforward network and the memorization component is implemented using a linear recurrence. The model update is computed as follows:

$$\begin{aligned} \boldsymbol{h}^t &= \sigma(\mathbf{W}_{xh}\boldsymbol{x}^t + \mathbf{W}_{mh}\boldsymbol{m}^{t-1}) \\ \boldsymbol{m}^t &= \mathbf{W}_{hm}\boldsymbol{h}^t + \mathbf{W}_{mm}\boldsymbol{m}^{t-1}, \end{aligned}$$

where $\boldsymbol{h}^t$ is the hidden state and $\boldsymbol{m}^t$ is the memory state. Despite the linearity of the recurrence, the model is equivalent to an RNN. To memorize the sequence of hidden activations $\boldsymbol{h}^1, \ldots, \boldsymbol{h}^T$ we can train the memorization component to encode the sequence. Since the memorization component is linear, it can be trained using the LAES algorithm [15] to obtain the optimal autoencoder of the hidden state sequence. Using this initialization technique, the memory component is able to encode the hidden state sequence, and therefore the memory can be

used to represent the entire sequence of extracted features. After the initialization, the model is finetuned with end-to-end training. In [2] we extended the *RNN pretraining* algorithm in [15], based on the LAES algorithm [15], to pretrain the LMN. In our experiments, we have found that the memory of the LMN, especially when initialized with the LAES, is superior to gated architectures when it comes to learning long-term dependencies. Table 1 shows the frame-level accuracy on the sequence modeling problem on four different MIDI datasets.

In a follow-up work, we extended the LMN by separating the memory components into k separate modules, each one taking the sequence of hidden activations with a different sampling rate. The resulting model, dubbed *Multi-Scale LMN*, is a hierarchical model inspired by the Clockwork LMN [13]. Hierarchical models are especially useful to process long sequences that contain long-term dependencies. The architecture of the Multi-Scale LMN can shorten the length of the dependencies between the input elements by subsampling the original sequence. The model is trained incrementally by adding a new memory module after a fixed number of epochs, each one initialized with the LMN pretraining algorithm. The experimental results improve on the state-of-the-art on the sequence generation and the common-suffix TIMIT [13] compared to equivalent Clockwork RNNs and LSTMs. Figure 1 shows a schematic view of the architecture of the LMN and the MultiScale LMN.

LMNs are related to another class of recurrent models: orthogonal recurrent networks. Imposing the orthogonality on the LMN and truncating the gradient, an approach inspired by the original LSTM training algorithm[11], the network gains a constant propagation of the gradient, and therefore the vanishing and exploding gradient problems are provably eliminated. However, compared to orthogonal models the LMN has two distinct advantages: first, despite the linearity of the memory, the entire network is still nonlinear since the functional component is a (possibly multi-layer) feedforward network. Orthogonal models instead can use a limited class of activation function to guarantee the constant gradient propagation. Furthermore, the pretraining with the LAES is much more effective than a random orthogonal initialization, since the resulting model is an optimal autoencoder. These advantages can be seen also in the experimental results, where the LMN achieves better results than any other orthogonal model in the literature on sequential MNIST and permuted MNIST. Other experimental results on TIMIT show a similar trend.

5 Conclusions and Future Work

The results of our research show that recurrent architectures can benefit from better training algorithms and initializations focused on solutions for the memorization subtask. The proposed conceptual separation allows separating the memory component to design novel models and training algorithms. An example of this approach is the Linear Memory Network, where the linearity of the memory is exploited to develop a pretraining algorithm that initializes the memory with the optimal autoencoder. The connection with the work on orthogonal

Table 1: Frame-level accuracy computed on the test set for each model. RNN-RBM results are taken from [4]

	JSB Chorales	MuseData	Nottingham	Piano MIDI
RNN	31.00	35.02	72.29	26.52
pret-RNN	30.55	35.47	71.70	27.31
LSTM	32.64	34.40	72.45	25.08
RNN-RBM*	33.12	34.02	**75.40**	**28.92**
LMN-A	30.61	33.15	71.16	26.69
LMN-B	33.98	35.56	72.71	28.00
pret-LMN-B	**34.49**	**35.66**	**74.16**	28.79

models is fundamental to guarantee good properties necessary to learn long-term dependencies. We remark how these results are a consequence of the separation of the model into two components.

In general, the experimental results show consistent improvements on several challenging datasets. The improvements are especially evident on datasets that require the memorization of long sequences, like complex sequences of notes in MIDI datasets or sequential pixel MNIST. These datasets are especially difficult for RNNs, which suffers from the vanishing gradient problem. They are also difficult for LSTMs, which tend to forget their input after a long sequence, due to the exponential effect of the forget gate. However, it must be noted that on different datasets, like several natural language processing benchmarks, the ability to forget past information seems a key component of every successful model. Therefore, we believe it is important to study new approaches that are able to combine the advantages of pure memorization models with architectures that are able to selectively forget.

This line of research offers several directions for future work. The linearity of the memory is useful to investigate the application of hessian-free optimization methods for RNNs.

Another possible line of research is the application of our approach to other fields. We are currently evaluating the domain of continual learning for sequential data as a possible avenue for future research. Continual learning models must be able to continually learn from new data without forgetting the old samples. We believe that this is a setting that could benefit from a separate memory, able to recognize the different samples and account for the differences between each subtask.

In conclusion, we believe a stronger focus on the memorization properties of recurrent models and training algorithms can bring large benefits to the field.

References

1. Arjovsky, M., Shah, A., Bengio, Y.: Unitary Evolution Recurrent Neural Networks (nov 2015), http://arxiv.org/abs/1511.06464

2. Bacciu, D., Carta, A., Sperduti, A.: Linear Memory Networks. In: ICANN (2019), http://arxiv.org/abs/1811.03356
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. CoRR **abs/1409.0**, 1–15 (2014). https://doi.org/10.1146/annurev.neuro.26.041002.131047, http://arxiv.org/abs/1409.0473
4. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. ICML (Cd) (2012), http://arxiv.org/abs/1206.6392
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR **abs/1412.3**, 1–9 (2014). https://doi.org/10.1109/IJCNN.2015.7280624, http://arxiv.org/abs/1412.3555
6. Elman, J.L.: Finding structure in time. Cognitive science **14**(2), 179–211 (1990)
7. Graves, A., Mohamed, A.r., Hinton, G.: SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton Department of Computer Science, University of Toronto. IEEE International Conference (3), 6645–6649 (2013). https://doi.org/10.1093/ndt/gfr624
8. Graves, A., Wayne, G., Danihelka, I.: Neural Turing Machines. CoRR **abs/1410.5**, 1–26 (2014). https://doi.org/10.3389/neuro.12.006.2007, http://arxiv.org/abs/1410.5401
9. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A.P., Hermann, K.M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., Hassabis, D.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471–476 (2016). https://doi.org/10.1038/nature20101, http://dx.doi.org/10.1038/nature20101
10. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02), 107–116 (1998)
11. Hochreiter, Sepp; Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1–32 (1997). https://doi.org/10.1144/GSL.MEM.1999.018.01.02
12. Johnson, M., Schuster, M., Le, Q.V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., Dean, J.: Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. In: TACL (nov 2017), http://arxiv.org/abs/1611.04558
13. Koutník, J., Ch, H., Greff, K., Ch, K., Gomez, F., Ch, T., Schmidhuber, U.: A Clockwork RNN. arXiv preprint arXiv:1402.3511 (2014), http://proceedings.mlr.press/v32/koutnik14.pdf
14. Mhammedi, Z., Hellicar, A., Rahman, A., Bailey, J.: Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections. In: ICML. pp. 2401–2409 (dec 2017), http://arxiv.org/abs/1612.00188
15. Sperduti, A.: Linear autoencoder networks for structured data. In: International Workshop on Neural-Symbolic Learning and Reasoning (2013)
16. Sperduti, A.: Equivalence results between feedforward and recurrent neural networks for sequences. IJCAI International Joint Conference on Artificial Intelligence **2015-Janua**(Ijcai), 3827–3833 (2015)
17. Vorontsov, E., Trabelsi, C., Kadoury, S., Pal, C.: On orthogonality and learning recurrent networks with long term dependencies. In: ICML. pp. 3570–3578 (jan 2017), http://arxiv.org/abs/1702.00071