ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Softmax Recurrent Unit: A new type of RNN cell

Lucas Vos[1] and Twan van Laarhoven[1,2]

1- Open University of the Netherlands - Faculty of Management, Science and Technology
Heerlen - The Netherlands

2 - Radboud University - Institute for Computing and Information Science
Nijmegen - The Netherlands

Abstract. Recurrent Neural Networks (RNNs) have been very successful in many state-of-the-art solutions for natural language tasks like machine translation. However, LSTM, the most common RNN cell, is complex and utilizes a lot of components. We present the Softmax Recurrent Unit (SMRU), a novel and elegant design of a new type of RNN cell. The SMRU has a simple structure, which is solely based around the softmax function. We present four different variants of the SMRU and compare them to both the LSTM and GRU on various tasks and datasets. These experiments show that the SMRU achieves competitive performance, surpassing either the LSTM or the GRU on any the given task, while having a much simpler design.

1 Introduction

Many machine learning tasks consist of dealing with sequential data, like machine translation, speech to text conversion, anomaly detection, or simply predicting the next value of a sequence. A Recurrent Neural Network (RNN) is the standard choice for these tasks because it captures sequential dependencies and can handle the variable input lengths seen in these datasets.

Ever since Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) unit [1] to overcome the vanishing gradients problem of plain RNN, the LSTM has achieved impressive results in many domains and tasks like handwriting recognition [2], machine translation [3] and speech recognition [4]. Although LSTM has been very successful, a major criticism is that it is an ad-hoc and complex architecture [5].

This criticism motivated us to look for a better architecture, especially one with a clear design and few components. The Softmax Recurrent Unit (SMRU) has an unsophisticated layout, completely built around the softmax function. We use the softmax function to control the hidden state, similar to the gates in LSTM and GRU, based on the input and previous hidden state. The result is a novel and elegant RNN unit designed around a single component. The idea to balance the gates with a single function forms a new class of RNN units.

Besides the basic SMRU, we introduce two small changes leading to four variants. These four variants of the SMRU are compared with the LSTM and GRU architectures on several tasks. The results of these experiments show that the more straightforward architecture of the SMRU achieves competitive performance to other architectures.

2 Related work

There has been some research into alternatives to the LSTM architecture.

In an effort to reduce the complexity of the LSTM architecture, Cho et al. [6] have introduced Gated Recurrent Unit (GRU). Compared to the LSTM the GRU has one fewer gate (3 instead of 4).

In 2018 Jozefowicz et al. performed an extensive search for RNN cells, to find alternatives to the LSTM [5]. Their scan involved a genetic-like algorithm constructing thousands of variants. Although some of these variants showed slightly better results on specific tasks, the conclusion was that the LSTM (along with the right bias initialization) gives very good results.

Greff et al. have also looked for alternative RNN cells, but their approach concentrated on a subset of 8 fixed LSTM variants to look at both the performance and the importance of hyper-parameters on larger datasets [7]. These variants did not perform significantly better on average, but they found that the learning rate is one of the most crucial hyper-parameters, and that the forget gate and output activation functions are the most critical of the LSTM cell.

Both works did not capture the architecture we present in Section 3, because only pointwise non-linearities where considered, and so the softmax function was not included in their scope.

3 The Softmax Recurrent Unit

We propose an alternative unit for RNNs, the Softmax Recurrent Unit (SMRU). Controlling this hidden state is key to an RNN cell because the hidden state must represent the crucial part of the information it has seen. We identify three important actions to manipulate the hidden state: *add* some value to it, *remove* some value and finally to *keep* a part of the previous state. These actions do not have to exclude each other and are mixed to compute a next hidden state. In the LSTM and GRU this is done through a complex set of gates. Instead we propose to use the softmax function,

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}.$$

So, at timestep t, in the SMRU the new hidden state is calculated as

$$\boldsymbol{h_t} = \text{softmax}(\boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{k}_t) \cdot (1, 0, \boldsymbol{h}_{t-1}) = \frac{\exp(\boldsymbol{a}_t) + \exp(\boldsymbol{k}_t)\boldsymbol{h}_{t-1}}{\exp(\boldsymbol{a}_t) + \exp(\boldsymbol{r}_t) + \exp(\boldsymbol{k}_t)},$$

using three gates activations

$$\begin{aligned}\boldsymbol{a}_t &= \boldsymbol{U}_a\boldsymbol{x}_t + \boldsymbol{V}_a\boldsymbol{h}_{t-1} + \boldsymbol{b}_a,\\ \boldsymbol{r}_t &= \boldsymbol{U}_r\boldsymbol{x}_t + \boldsymbol{V}_r\boldsymbol{h}_{t-1} + \boldsymbol{b}_r,\\ \boldsymbol{k}_t &= \boldsymbol{U}_k\boldsymbol{x}_t + \boldsymbol{V}_k\boldsymbol{h}_{t-1} + \boldsymbol{b}_k,\end{aligned}$$

for the $a$dd, $r$emove and $k$eep actions. Here $\boldsymbol{U}$ and $\boldsymbol{V}$ are weights on the input and on the previous hidden state respectively, and the $\boldsymbol{b}$ are bias terms.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

We use the state $\boldsymbol{h}_t$ also as the output of the SMRU. Note that the hidden state always lies between 0 and 1. By increasing $\boldsymbol{a}_t$ the new hidden state becomes closer to 1, by increasing $\boldsymbol{r}_t$ the hidden state becomes closer to 0, and by increasing $\boldsymbol{k}_t$ the hidden state stays closer to the previous state.

3.1 Variants

After the introduction of the LSTM, additional changes were introduced to the architecture [8] [7]. Inspired by the history of the LSTM, we look at different SMRU implementations. We introduce two changes of the basic architecture which we believe are natural extensions.

Centered variant (SMRU-c) It has been previously noted that the RNN cells benefit from using state centered around 0. To achieve this in the SMRU, we give the remove gate an active role in the final calculation of the hidden state,

$$\boldsymbol{h_t} = \text{softmax}(\boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{k}_t) \cdot (1, -1, \boldsymbol{h_{t-1}}).$$

This extends the range of the hidden state from $[0..1]$ to $[-1..1]$.

Softmax first (SMRU-s) In this variant, we perform the softmax function before the addition of the weighted input and weighted previous hidden state. The inspiriation comes from the MUT1 cell considered in [5], which showed very good results by using a similar idea of early non-linearties.

In this variant we split the gates in two, one part based on the input and one part based on the previous state, to get

$$\boldsymbol{a}_{tx} = \boldsymbol{U}_a \boldsymbol{x}_t + \boldsymbol{b}_{ax} \qquad \boldsymbol{a}_{th} = \boldsymbol{V}_a \boldsymbol{h}_{t-1} + \boldsymbol{b}_{ah},$$

and similarly for the remove and keep gates. The new hidden state of this variant is then calculated by

$$\boldsymbol{h_t} = \text{softmax}(\boldsymbol{a}_{tx}, \boldsymbol{a}_{th}, \boldsymbol{r}_t, \boldsymbol{r}_{th}, \boldsymbol{k}_{tx}, \boldsymbol{k}_{th}) \cdot (1, 1, 0, 0, \boldsymbol{h}_{t-1}, \boldsymbol{h}_{t-1})$$

We also consider a SMRU-cs variant that combines both ideas.

3.2 Initialization

Gers et al. discovered that the performance of the LSTM increases when the bias of the forget gate is set to 1 [9]. Jozefowicz et al. later reconfirmed this effect [5].

This discovery inspired us to search for bias initialization effects on the SMRU. We will look at three bias initializations variants, *b-add*, *b-remove*, *b-keep*, which initialize the corresponding named gate at 1 while holding the others at 0. We will compare these to the default zero initialization (*b-zero*).

4 Experiments

We have implement the SMRU in PyTorch 1.0.1. The code is available on GitHub[1].

[1]https://github.com/voslucas/smru

	mnist56	**ptb**	**luong**	**bahdanau**	**nott1**	**nott2**
model	↑ acc	↓ ppl	↑ bleu	↑ bleu	↓ nll	↓ nll
LSTM	89.90	**97.49**	22.76	**24.10**	3.52	3.48
GRU	89.70	101.57	**23.04**	23.12	3.55	3.29
SMRU	88.20	106.85	22.74	22.95	**3.42**	3.31
SMRU-c	**91.95**	n/a	21.56	22.31	3.45	**3.28**
SMRU-s	81.50	108.55	22.20	23.52	3.77	3.55
SMRU-cs	90.40	99.07	22.55	23.51	3.83	3.31

Table 1: Results of the tasks per RNN cell. (↑ higher is better, ↓ lower is better)

We compare the performance of the SMRU variants with LSTM and GRU on four different datasets. We only change the RNN cell in these tasks, and keep all other model and training parameters the same. The details of the used network architectures and training parameters are omitted for space reasons. These can be found in the source code.

The datasets we use are:

MNIST56: The MNIST dataset contains 60000 training images and 10000 test images of 28x28 pixels representing the digits 0 to 9. We divide the pixels of each image from top left to bottom right into a 56 step long sequence of 14 pixel values. After the sequence, the network needs to predict the correct digit.

PTB: Penn TreeBank is a large annotated corpus of English text [10]. The task is to predict the next word given a context of previous words. We use the pre-processed dataset of Mikolov et.al. [11].

IWSLT: The International Workshop on Spoken Language Translation '15 English-Vietnamese is a small dataset of 133k sentence pairs. The task is to translate the sentence. We reuse the codebase of JoeyNMT toolkip [12], as well as the supplied model and hyper-parameters. We use SacreBLEU as our scoring mechanism for comparison between the different cell types [13]. We used two predefined models coming from the Joey NMT toolkit [12], one using Bahdanau attention [14] and the other with Luong attention [15].

Nottingham: A collection of 1200 British and American folk tunes. The original chords are transformed into 88 binary notes (0=off, 1=on) and every timestep represents an eighth note. The goal is to predict the next note given a list of played notes. The performance is measured in terms of negative log-likelihood (NLL) loss. We use the dataset with two models: one uses a single layer (nott1) and the other uses two layers of RNN cells (nott2).

5 Results

The results of the experiments are shown in Table 1. The SMRU design shows very competitive results in comparison to the GRU and LSTM.

In the Notthingham and MNIST56 datasets, one of the SMRU variants achieves the highest scores.

	mnist56	**ptb**
model	↑ acc	↓ ppl
b-zero	81.25	107.58
b-add	79.30	111.56
b-remove	74.65	109.12
b-keep	**88.20**	**106.84**

Table 2: Comparison of bias initialization methods. All with the base SMRU variant. (↑ higher is better, ↓ lower is better)

In both language-oriented tasks (PTB and IWSLT), the SMRU did not gain first position, however, the SMRU shows similar or better performance than either the LSTM or the GRU. We note that the design of the experiments is a simple change of the RNN cell, leaving all hyper-parameters the same and no RNN cell specific tuning is performed. The language task models and parameters where optimized around the LSTM and GRU. Higher scores might be possible when the hyper-parameters are optimized for the SMRU.

Of the SMRU variants, the original architecture shows the best overall performance, but it falls behind on the PTB en Bahdanau task. The SMRU-c (centered) achieves two first positions in the overall task list (on MNIST56 and NOTT2), however on the PTB task, the SMRU-c failed to converge. This might be due to the higher effective learning rate caused by doubling the range of state values. The SMRU-s (softmax-first) variant has the worst overall performance. The combination SMRU-cs is the most stable over all tasks.

We have also performed experiments on the different bias initialization methods on the MNIST56 and PTB tasks, see Table 2. Here we consistently find that initializing the bias of the keep gate to 1 (b-keep) gives the best results. This matches previous results about the forget gate of LSTM. Likely because it means that more of the previous state is kept, increasing the gradient flow to earlier timesteps. The other bias initialization schemes perform worse than zero initialization. All other experiments were performed with his b-keep initialization.

6 Conclusion

We have introduced a new type of RNN cell, the Sotmax Recurrent Unit (SMRU). The architecture of the SMRU cell is based solely around the softmax function. The use of a softmax function gives an elegant way of combining multiple gates in a single step. This design not considered by previous searches of LSTM variants, and using softmax or similar functions could open up new search space for network architecture searches.

We found that the SMRU benefits from a keep-gate bias initialization of 1. Besides the basic design of the SMRU, we have tested some variants. The SMRU-c and the SMRU-cs variants show the most potential whenever hyper-parameter tuning is an option, because they are very sensitive to the used learning rate. In comparison to the LSTM and GRU, the SMRU variants achieve better performance than either the GRU or the LSTM within the given tasks.

In our experiments we have used a CPU-based implementation because of its flexibility. This limits the applicability to large datasets and models. However there is no technical reason why a GPU-based could not be made.

References

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

[2] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, May 2009.

[3] Minh thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *In ACL*, 2015.

[4] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[5] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct 2017.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, Sep. 1999.

[9] Felix A Gers and Jurgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE, 2000.

[10] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. 1993.

[11] Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Honza Černocký. Empirical evaluation and combination of advanced language modeling techniques. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 605–608, 2011.

[12] Julia Kreutzer, Joost Bastings, and Stefan Riezler. Joey NMT: A minimalist NMT toolkit for novices. *To Appear in EMNLP-IJCNLP 2019: System Demonstrations*, Nov 2019.

[13] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[15] Minh-Thang Luong and Christopher D Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, 2015.