ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

Surprisal-Based Activation in Recurrent Neural Networks

Tayfun Alpay, Fares Abawi and Stefan Wermter

University of Hamburg - Department of Informatics
Vogt-Koelln-Str. 30 - 22527 Hamburg - Germany
http://www.informatik.uni-hamburg.de/WTM

Abstract. Learning hierarchical abstractions from sequences is a challenging and open problem for Recurrent Neural Networks (RNNs). This is mainly due to the difficulty of detecting features that span over long distances with also different frequencies. In this paper, we address this challenge by introducing surprisal-based activation, a novel method to preserve activations contingent on encoding-based self-information. The preserved activations can be considered as temporal shortcuts with perfect memory. We evaluate surprisal-based activation on language modelling by testing it on the Penn Treebank corpus and find that it can improve performance when compared to a baseline RNN.

1 Introduction

Recurrent Neural Networks (RNNs) are powerful sequence processing models that are equipped with memory from recurrent feedback connections. While they are naturally unable to learn long-term dependencies due to vanishing gradients, this issue was addressed by the introduction of gating units, made popular by the Long Short-Term Memory (LSTM; [3]). The success of gating models has led to a wide range of practical applications and a generally increased interest and research on RNNs.

One drawback of these models is that they operate in discrete time steps and also update at every time step. This makes it generally difficult to learn temporal features that have a significantly different resolution than their input frequency. We hypothesize that the capability to learn *when* to update could open up promising directions for a number of current research problems, such as dealing with event-driven extremely long sequences (e.g. raw audio data) without preprocessing, having adaptive computation times [2], learning multiple timescales and their boundaries [1, 5], or integrating sensory data under asynchronous sampling conditions [9].

Some recent approaches focus on the idea of suppressing hidden unit activations under specific conditions. The recently proposed Clockwork RNN (CWRNN; [5]) utilizes a hidden layer that is partitioned into *modules* which are only activated on specific timesteps. Inactive modules simply preserve their hidden activation from the previous timestep until a periodic clock sets them active again. The benefit is that the network has perfect memory along inactive paths. A large disadvantage of the CWRNN, however, is that the periodic activation conditions are i) predefined and not learned, and ii) global for the

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

entire sequence. This can lead to challenges when dealing with varying distances between dependencies or phase shifts which are common in real-world applications. This idea has been developed further as a regularization method called zoneout [6]. Zoneout randomly preserves the previous activations of hidden units and can therefore be seen as a variant of dropout in which connections are masked with a random mask of ones (copy) instead of zeros (drop). In the context of recurrent architectures, zoneout is also a special case of the CWRNN with random clocking and a single module. The Phased LSTM [9] extends the LSTM with adaptive time gates that can learn when to preserve and when to activate based on rhythmic oscillations. It introduces adaptive preservation but has a number of new parameters that have to be tuned or learned. Another report proposes to perform zoneout based on the information-theoretic surprisal from feeding back the prediction error [7]. However, while this is fully adaptive, it depends on additional supervised information in both training and test phases. Surprisal (also called self-information) has also previously been used successfully in natural language processing as a preprocessing metric for segmentation [4].

In this paper, we introduce a novel activation mechanism based on surprisal that is in itself fully unsupervised, i.e. depends neither on labelled data nor prediction errors. Instead of applying pre-/postprocessing on in-/output, we observe the surprisal *directly* on the encoding and preserve states locally within modules as long as the surprisal does not shift significantly.

2 Surprisal-based Activation

In the following, we assume a standard RNN with a single hidden layer $\mathbf{h}_t$, of which the candidate activation $\hat{\mathbf{h}}_t$ is calculated as follows[1]:

$$\hat{\mathbf{h}}_t = f(\mathbf{W}_{xh} \cdot x_t + \mathbf{W}_{hh} \cdot \mathbf{h}_{t-1}),$$

where $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ are the input and recurrent weight matrices, and $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ the input and previous state, respectively. We further partition our hidden layer into M evenly-sized modules $\mathbf{m}_t^{(i)}$ such that $\hat{\mathbf{h}}_t = [\hat{\mathbf{m}}_t^{(1)}; \hat{\mathbf{m}}_t^{(2)}; \cdots ; \hat{\mathbf{m}}_t^{(M)}]$. The purpose of these modules is to introduce stability through ensembling since the decision to apply the candidate activation $\hat{\mathbf{h}}_t$ or to preserve $\mathbf{h}_{t-1}$ is made per module. We therefore pool the candidate activations in the next step to compress the activations on a per-module-basis:

$$\mathbf{p}_t = g(\mathbf{m}_t^{(i)})$$

where $\mathbf{p}_t$ is the resulting pooling vector with $|\mathbf{p}_t| = M$ and $g(\cdot)$ is a pooling operator such as max or avg. In the next step, we normalize with softmax σ and calculate the surprisal from the resulting probability distribution:

$$s_t = log\Big(\frac{1}{\sigma(\mathbf{p}_t)}\Big) = log\Big(\frac{\sum_{i=1}^{M} exp(\mathbf{p}_t^i)}{exp(\mathbf{p}_t)}\Big)$$

[1]To apply surprisal-based activation on an LSTM or GRU, one can apply their respective gated update rules to $\hat{\mathbf{h}}_t$ instead.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

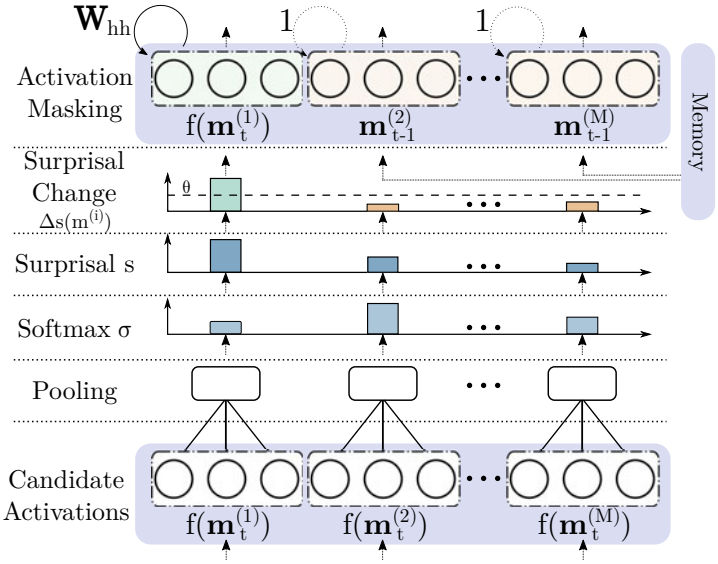

Fig. 1: Overview of surprisal-based activation within a hidden layer. Module activations in t that are significantly different to the previous timestep $t-1$ are kept (green), all other modules preserve their previous states $\mathbf{m}_{t-1}^{(i)}$ (red).

As a final step, each module $m_t^{(i)}$ is activated depending on their respective change in surprisal. We choose candidate activations where the surprisal change is larger than an experimentally determined hyperparameter θ, and otherwise preserve all states:

$$\mathbf{m}_t^{(i)} = \begin{cases} \hat{\mathbf{m}}_t^{(i)} & \text{if } |s_t - s_{t-1}| > \theta, \\ \mathbf{m}_{t-1}^{(i)} & \text{otherwise.} \end{cases}$$

Fig. 1 illustrates our complete approach. All other parts of the RNN are trained normally and backpropagation through time is executed without any modifications. We hypothesize that the combination of local pooling and preservation of previous states based on surprisal does not only maintain the regularizing effect of zoneout, but can also lead to a self-organization process in which modules separately learn features over long time distances during which the input, and subsequently the encoding, does not undergo significant changes.

One potential pitfall of masking activations by internal surprisal is a "deadlock" in which an encoding sequence with mostly constant surprisal is never updated. To counteract this, we element-wise multiply the hidden state with a decay mask:

$$\hat{\mathbf{h}}_t = \hat{\mathbf{h}}_{t-1} \odot \mathbf{d}_t \text{ with } \mathbf{d}_t = (P(\mathbf{d}_t^{(1)}), \cdots, P(\mathbf{d}_t^{(|\mathbf{h}|)}))$$

where $\mathbf{d}_t$ is a decay vector, and $P(d_j = \alpha) = p_d$ and $P(d_j = 1) = (1 - p_d)$ are empirically determined probabilities to decay the activation of unit j by $\alpha = 0.01$

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

Table 1: Single model validation and test perplexity (the lower, the better) of our models (+S) compared against the baseline on the Penn Treebank dataset.

Model	Best Val.	Test
RNN	130.4	140.8
RNN+S	131.5	**126.4**
LSTM	123.6	**121.5**
LSTM+S	124.3	123.1

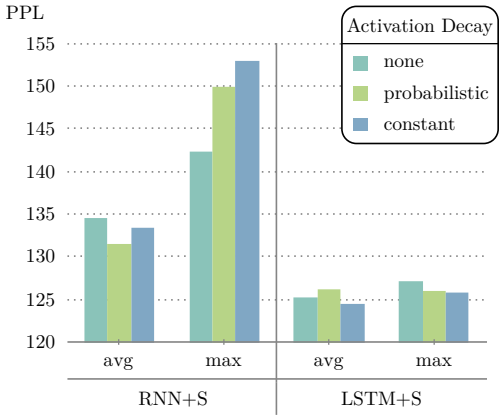

Fig. 2: Comparing decay and pooling methods from our approach by validation perplexity.

with probability $p_d = 0.2$. This introduces variability on sub-module level and allows single units to counteract potential information loss from pooling.

3 Evaluation

3.1 Experimental Setup

To investigate the effect of surprisal-based activation, we evaluate our model with language modelling on the Penn Treebank Corpus, a relatively small but well-known benchmark. While many recent studies make heavy use of regularization techniques to combat the corpus' high susceptibility for overfitting and reach the lowest possible perplexity (PPL), we are mainly concerned with evaluating performance gains from our approach in controlled conditions. We therefore run our own RNN and LSTM models as a baseline and apply our method under the same experimental conditions. Our hyperparameters for gradient descent follow (where possible) the previous state-of-the art for this dataset, achieved by Recurrent Highway Networks [11], even though we restrict ourselves to a medium-sized layer with 1000 hidden units. We run 100 configurations and repeat every experiment three times. We test variations for pooling (max vs. average), activation decay (none, probabilistic, constant), number of modules M (1, 2, 4, 8), and activation function (sigmoid, tanh). Training stops at 20 epochs due to risk of overfitting and we record the epoch with the best validation perplexity. After selecting the best models by validation perplexity, we then run the evaluation on the test set.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

M	1	2	5	10	25	50	100	250	500	1000
Valid.	681.4	136.0	136.1	**133.8**	134.2	134.0	135.9	154.9	299.2	312.2
Test	634.7	130.2	130.4	**127.6**	128.4	128.8	130.3	148.2	280.6	296.7

Table 2: Validation and test perplexities for different number of modules M in the RNN+S.

3.2 Overview

Table Tab. 1 shows the overall results comparing our lowest achieved perplexities against the baselines. As can be seen, surprisal-based activation in an RNN (RNN+S) yields a significantly better generalization on the test set when compared to the baseline (RNN). It is important to note that an unregularized RNN language model can be trained to test perplexities as low as 124.7 [8], which in turn would also improve our RNN-specific hyperparameters, and consequently also our RNN+S model. The baseline LSTM yields a better overall score although surprisal (LSTM+S) does not quite improve the language model. A possible explanation could be that the unit-level gates slightly counteract the module-level effect of surprisal.

3.3 Pooling and Decay

The impact of pooling and decay can be seen in Fig. 2. Average pooling yields overall better perplexities. The negative effect of max pooling is, however, somewhat reduced for the LSTM+S, most likely due to a low numerical difference between the average activations in each module and their maximum. The decays also have an adverse effect when applied to the LSTM. The RNN+S benefits the most from applying a unit-level decay mask (see Sec. 2) when using average pooling. A constant, global decay gives slightly worse results, on par with having no decay at all, indicating that a decay on sub-module level is indeed the best choice. This gives us evidence that random variations on a lower granularity than the pooling process might help with any surprisal-based "deadlocks" (see Sec. 2).

3.4 Number of Modules

Deadlocks from constant surprisal are especially significant when the mask is applied to the entire layer, as is the case with $M = 1$. Indeed, the RNN+S produces significantly worse results whenever the hidden layer is not partitioned into modules (643.7 PPL). This is confirmed by the performance increasing with the number of modules (128.6 PPL for $M = 2$, 127.9 PPL for $M = 4$, 126.4 PPL for $M = 8$). Interestingly, we observed that the LSTM+S is quite stable under $M = 1$, most likely as a side effect of the gating units. To further investigate the ideal partition size, we ran additional trials for the best RNN+S model with $M \in \{10, 25, 50, 100, 250, 500, 1000\}$. As can be seen from Tab. 2, performance increases until M is increased to 10 but reduces for additional modules after this point. Since less modules equate to more units per module, the performance

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

drop for $M < 10$ can additionally be explained by the larger area of effect of pooling. However, the opposite end of the spectrum, such as for $M = 1000$ (where each module consists of a single unit) shows that the absence of pooling worsens the language models considerably. This gives us evidence about the importance of modules when pooling, and more generally, how balancing the correct degree of locality influences binary activation masking.

4 Discussion

We have introduced a novel method to preserve activations in RNNs based on surprisal. Our results show that surprisal-based activation can improve generalization for RNN language models. When applied to gated units, however, we have observed a slightly mitigated effect, most likely due to the LSTM having a higher variability on unit-level as a result of its gates. We therefore hypothesize that the surprisal mechanism should be implemented directly as part of the gating process for the LSTM. The next step is to integrate both approaches. Future work will also start with investigating the internal dynamics that emerge from our approach. This would also be facilitated by an evaluation of character-level language modelling and other tasks with hierarchical temporal dependencies. Future work will also include an investigation into surprisal-based attention as both surprisal and attention are suspected to have related roles for language processing within the predictive coding framework [10].

References

[1] T. Alpay, S. Heinrich and S. Wermter, Learning multiple timescales in recurrent neural networks. In Proc. *ICANN*, LNCS 9886, pages 132–139, Springer Verlag, 2016.

[2] A. Graves, Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983, 2016.

[3] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10):2222-2232, 2017.

[4] S. S. Griffiths, M. M. McGinity, J. Forth, M. Purver and G. A. Wiggins, Information-theoretic segmentation of natural language. In Proc. *AIC*, pages 54–67, 2015.

[5] J. Koutník, K. Greff, F. Gomez and J. Schmidhuber, A clockwork rnn. In Proc. *ICML*, pages 1863–1871, 2014.

[6] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. Courville and C. Pal, Zoneout: Regularizing rnns by randomly preserving hidden activations. arXiv preprint arXiv:1606.01305, 2016.

[7] K. Rocki, T. Kornuta and T. Maharaj, Surprisal-driven zoneout. Tech report, arXiv preprint arXiv:1610.07675, 2016.

[8] T. Mikolov, Statistical language models based on neural networks. PhD thesis, Brno University of Technology, 2012.

[9] D. Neil, M. Pfeiffer and S. C. Liu, Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In Proc. *NIPS*, pages 3882–3890, 2016.

[10] A. Zarcone, M. V. Schijndel, J. Vogels and V. Demberg, Salience and attention in surprisal-based accounts of language processing. Frontiers in Psychology, 7:844–860, 2016.

[11] J. G. Zilly, R. K. Srivastava, J. Koutník and J. Schmidhuber, Recurrent highway networks. arXiv preprint arXiv:1607.03474, 2016.