BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Randomly Sparsed Recurrent Neural Networks in Language Modeling

Karel Beneš*

Abstract
The models based on recurrent neural networks are the current state of the art method in statistical language modeling. Despite the recent major advances, the task of training a well performing network is still difficult. This work investigates performance of recurrent network with tightly constrained weight matrix, where a large portion of randomly picked recurrent weights is forced to be zero. The effect of decreasing the number of parameters on the performance of the model is studied. Additionaly, the model combinations are investigated. Although the proposed architecture did not achieve any improvement over the baseline model, the combination of the sparse models performs better than the combination of fully connected RNNs.

Keywords: Sparse weights matrix — Recurrent neural network — Statistical language modeling

Supplementary Material: Downloadable Code

*xbenes20@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The statistical modeling is a problem of finding such a model, which would assign high probabilities to sentences which are frequent in a given natural language, while the sentences that are rare or plain invalid are assigned low probability. A high quality language model is an important part of systems for automatic speech recognition or text compression algorithms.

Since their introduction [1] to the field of statistical language modeling, the recurrent neural networks (RNN) have been defining the state of the art. On top of the original architecture, substantial improvements have been reached by using more complex architectures, such as Long Short-Term Memory [2]. On the other hand, a recent work by Mikolov et al. [3] suggests, that improvements of similar magnitude may be obtained by much simpler techniques.

At the same time, also a more theoretically motivated work appeared [4], advocating the sparse initialization of the recurrent weights.

In this paper, a rather practically formulated approach to enforcing the sparsity is studied.

To compare performance of different models, the per-word entropy is used. Let w_i denote the i-th word and w_f^t the sequence of words $w_f w_{f+1} \dots w_t$, and let the probability $p(\text{current word}|\text{history})$ be defined by the examined model. Then the per-word entropy of a given text is defined as:

$$\text{per-word entropy} = -\frac{1}{N}\sum_{i=1}^{N} \log_2 p(w_i|w_1^{i-1}) \qquad (1)$$

The per-word entropy relates directly to the number of bits needed to store the test sentence, if we use the given model for compression. The lower – the better.

2. Previous Work related to the Sparsity of the Recurrent Weights Matrix

Let the $\mathbf{x}_t$ denote the input vector at time t and let $\mathbf{y}_t$ be the output vector at time t, regardless of the specific architecture of the language model. The input vector is word w_t encoded as 1-of-K. Similarly, the expected output is the word w_{t+1}, encoded as 1-of-K. The actual output of the network is a vector of probabilities which sums up to one.

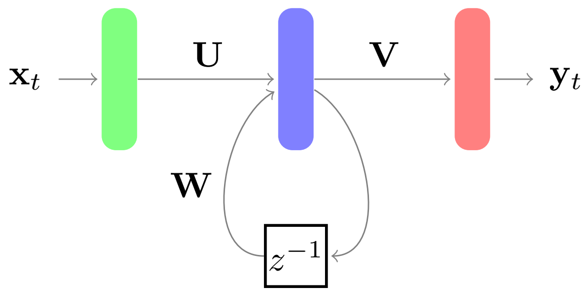


Figure 1. Diagram of a Simple Recurrent Network. The z^{-1} block represents a one timestep delay. This was the first recurrent neural network model introduced into language modeling. The work presented in this paper is based directly on it.

The simple recurrent network (SRN) [5] is defined by these two equations:

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1}) \tag{2}$$

$$\mathbf{y}_t = \text{softmax}(\mathbf{V}\mathbf{h}_t) \tag{3}$$

Here, the $\mathbf{h}_t$ is a compact representation of history, including the current word. Matrix $\mathbf{W}$ holds the recurrent weights – this paper deals with altering it.

The enhanced Structurally Constrained Recurrent Network model of Mikolov et al. adds neurons dedicated to preserving longer-time memory:

$$\mathbf{s}_t = (1-\alpha)\mathbf{B}\mathbf{x}_t + \alpha\mathbf{s}_{t-1} \tag{4}$$

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{P}\mathbf{s}_t) \tag{5}$$

$$\mathbf{y}_t = \text{softmax}(\mathbf{V}_h\mathbf{h}_t + \mathbf{V}_s\mathbf{s}_t) \tag{6}$$

With $\alpha \in (0;1)$, the "slower" neurons $\mathbf{s}_t$ represent an exponentially decaying sum of all previous words, as projected by matrix $\mathbf{B}$.

The recurrent computation of h_t, s_t may be viewed as a projection by a single matrix $\mathbf{R}$. Matrix $\mathbf{R}$ has large parts set to zero, as illustrated in Eq. (7). Hence the name Structurally Constrained RNN. At this point, it is necessary to point out that in this approach, the hidden layer is not processed uniformly, as the logistic sigmoid nonlinearity is applied only to the "fast" neurons.

$$\mathbf{R} = \left[\begin{array}{c|c} \mathbf{W} & \mathbf{P} \\ \hline \mathbf{0} & \alpha\mathbf{I} \end{array}\right] \tag{7}$$

There are other points of view advocating sparseness of the recurrent matrix:

So called Echo State Networks (ESN) have been shown to predict timeseries quite well [6]. An ESN is an RNN, where the input and recurrent connections are fixed, only the output weights are learned. Moreover, the recurrent weights are fixed as sparse, e.g. only 15 input connections for every neuron in a hidden layer wide several hundreds of neurons.

In their recent paper, Sutskever et al. have studied [4] the impact of different random initialization of the recurrent weights on the behaviour of the network. They have approached the problem from the ESN and suggested that initializing the network's recurrent connections as sparse helps to learn the longer-term dependencies.

Yoshua Bengio et al. promote a different approach, forcing the activations to be sparse [7]. They do so by using rectified linear units as nonlinearities and applying an L1 regularization on the activations. This allowed them to obtain an improvement over an RNN enhanced by neurons acting as leaky integrators.

3. Studied Architecture of Randomly Sparsed Recurrent Neural Network

I take the Simple Recurrent Network as a baseline and change the computation of the hidden state, i.e. replacing the Equation (2) with (8).

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + (\mathbf{W}_m \odot \mathbf{W})\mathbf{h}_{t-1}) \tag{8}$$

The $\odot$ represents element-wise product, also known as Hadamard or Schur product. In (8), I introduce the *mask* $\mathbf{W}_m$ of the recurrent weights matrix $\mathbf{W}$. The mask is a square matrix of same the shape as $\mathbf{W}$ and $(w_{m_{ij}}) \in \{0,1\}$. Thus, the product $\mathbf{W}_m \odot \mathbf{W}$ forces some of the weights in $\mathbf{W}$ to be zero.

It can also be viewed as an ESN with non-zero weights turned learnable, or a sparsely initialized RNN, where the initial sparsity constraints have been kept throughout the training.

In the scope of this paper, I investigate the behaviour of such networks, that have the mask $\mathbf{W}_m$ fixed. Therefore, the $\mathbf{W}_m$ is not a learnable parameter of the model, unlike $\mathbf{W}$. During the experiments, the elements of the mask are sampled from a Bernoulli distribution. The parameter p of the distribution is a tunable hyper-parameter. Also, this hyper-parameter can be interpreted as the density of the mask.

I call the resulting model a Randomly Sparsed Recurrent Neural Network (RS-RNN).

In order to allow the neurons to remember their individual values, the diagonal of the mask is always set to one, increasing slightly the overall density of the mask.

3.1 Implementation

The computational model is implemented using the Theano toolkit [8] [9]. For learning of the parameters,

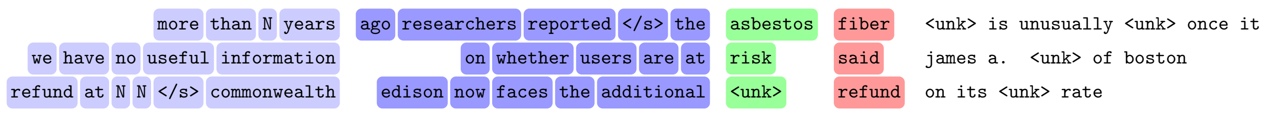

more than N years ago researchers reported </s> the asbestos fiber <unk> is unusually <unk> once it
we have no useful information on whether users are at risk said james a. <unk> of boston
refund at N N </s> commonwealth edison now faces the additional <unk> refund on its <unk> rate

Figure 2. Multistream learning with 3 parallel streams. Red words are the targets, green words are the current input. Backpropagation through time is done over the five dark blue words, the light blue words only affect the current learning step by being encoded in the hidden state of the network.

I used the stochastic gradient descent with backpropagation through time (BPTT).

I have implemented the *multistream learning*. In this learning scheme, illustrated in Figure 2, there are more floating windows on the training corpus and error is computed simultaneously on all of them. In the preliminary experiments, I found out that using more than 4 streams for the training hurts the performance. This can be explained by the significant decrease of number of updates per epoch. Therefore, only 4 stream are used for training of networks in following experiments.

As usual with the experiments involving random initialization, the implementation contains the seed, which is used for random number generation. It has two separate seeds: one for the mask and the other for the weights.

The weights are updated once every 30 forward steps. While the error from the last words of these sequences is propagated over whole these sequences, a BPTT over 5 timesteps is guaranteed even for the first word. This is achieved by overlapping the sequences.

I do not use any form of hierarchical softmax for computing the output.

4. Performance on the Penn Treebank Corpus

Penn Treebank corpus (PTB) is a subset of the Wall Street Journal corpus[1]. It has been hand-annotated for grammar categories at University of Pennsylvania, thus its name. For PTB, there is a vast comparative publication available [10].

There is also a widely used preprocessed version[2], where the vocabulary is reduced to 9999 most common words and the rest is replaced by an `<unk>` token.

The widely used version of PTB is divided into three parts: Training set consists of approx. 890 thousands tokens in approx. 42 000 sentences. Validation subset consists of approx. 70 000 tokens in 3370 sentences. Similarly, the test set contains approx. 79 000 tokens in approx. 3700 sentences.

The SRN architecture with respective number of hidden units is taken as the baseline. The experiments are aimed at performance of the network with respect to the density of the random mask. Results of the first set of experiments are captured in Figure 3a. Since there is principally more randomness in the RS-RNN than in a regular RNN, I run every experiment 9 times, using 3 different seeds for sampling the mask and 3 for sampling the weights.

The results show several findings: With sparser mask, the difference between performance on training and validation/test sets increases. The improvement on the training set is of greater magnitude than the decrease on unseen data, but the variation of the performance of models with different random initializations increases as well. Overall, the decrease of performance on unseen data is rather small, given we reduce the number of recurrent connections down to 20 %.

Next, scaling of the RS-RNN is examined. For this, twice as big hidden layer was used, i.e. it had 200 hidden units. Again, nine experiments with different random seeds are conducted for each density of the random mask, results are captured in Figure 3b.

The overall behaviour of RS-RNN with respect to the density of random mask is intact with wider hidden layer. However, the differences emerging from different density are of a greater magnitude. The results for each density are more variable as well.

4.1 Interpolation of several models

It follows from the initial experiments, that the RS-RNNs tend to improve on the training set as the density of the random mask drops. So I next examined how do these models combine. I used the simple linear interpolation with no weighting:

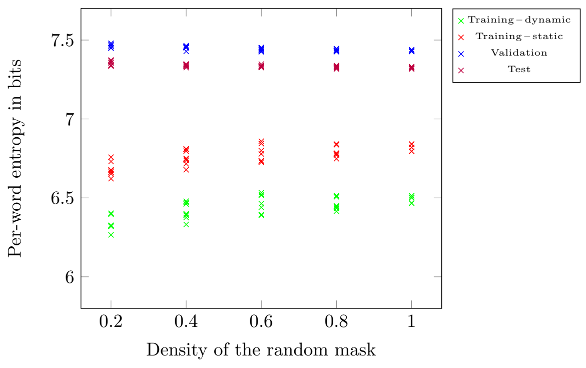
$$p(w_t|p_1^{t-1}) = \frac{1}{N}\sum_{i=1}^{N} p_i(w_t|p_1^{t-1}) \qquad (9)$$

I have compared the interpolation of nine models to the average of their individual errors, as show in Figure 4. It is apparent that the interpolation of models of same architecture provides a significant improvement over using these models separately. This is already a well known result.

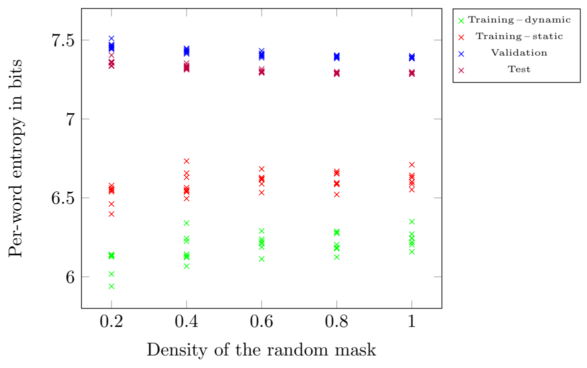
A novel observation lies in the gain from interpolation as conditioned on density of the mask. Trivial fact

[1]LDC item number LDC99T42, https://catalog.ldc.upenn.edu/LDC99T42

[2]http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz



(a) Networks with 100 hidden units.



(b) Networks with 200 hidden units.

Figure 3. A basic performance overview of models with different density of the mask $\mathbf{W}_m$. For every examined density, nine experiments with different seeds were performed. The networks with density 1 are equivalent to a regular SRN. The error reported as "Training – dynamic" is obtained by accumulating the errors during training, i.e. when the network is adapting its weights to the text. Rest of the statistics is computed without any adaptation.

Table 1. Per-word entropy of the posterior combination of 9 models, as dependent on density of the random mask. All models have 200 hidden neurons.

	Density of the random mask				
Dataset	0.2	0.4	0.6	0.8	1
Average					
Validation	7.46	7.42	7.40	7.39	7.38
Test	7.35	7.32	7.30	7.28	7.28
Interpolation					
Validation	7.10	7.08	7.09	**7.07**	7.14
Test	7.00	6.99	6.99	**6.97**	7.05

is, that the gain increases as the average performance of sparser models is slightly worse. However, it is consistent over both model sizes investigated, that it significantly helps to interpolate at least slightly sparsed models.

It is likely, that the RS-RNN models combine better because having less parameters, they are forced to focus on a subset of features in the text. And since the subset is randomly picked, it is likely that more such subsets will be complementary. It remains to discover, why the significance of the effect is not increasing as the mask is getting sparser.

5. Conclusions

In this work, I have proposed the Randomly Sparsed Recurrent Neural Network architecture and studied its basic performance properties. The proposed model is a logical continuation of several independently designed language models based on RNNs.

Making the recurrent weights sparse brings little harm on validation error, while it helps the model to learn the training corpus. This is counter-intuitive, as we usually observe precise opposite when decreasing the number of parameters. On the other hand, the RS-RNN has been shown to have greater potential for model combination than the dense SRN architecture.

A continuation of the work is to examine similar approach to make the word embeddings sparse and finally making these two work together. The presented work is a part of my diploma thesis.

Acknowledgements

I would like to thank my supervisor Mirko Hannemann for his help.

References

[1] Tomáš Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, 2012.

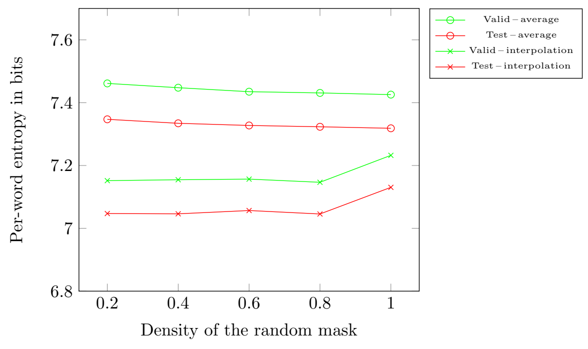
[2] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, OR, USA, September 2012.

[3] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2014.

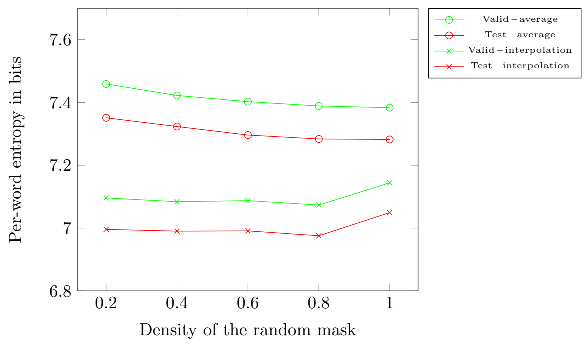
[4] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning.

[5] Jeffrey L. Elman. Finding Structure in Time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[6] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press, 2003.



(a) Networks with 100 hidden units.



(b) Networks with 200 hidden units.

Figure 4. Comparison of linear interpolation of nine model with same architecture and different random initializations to the averaging of their individual performance.

[7] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in Optimizing Recurrent Networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628, May 2013.

[8] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[9] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[10] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky. Empirical evaluation and combination of advanced language modeling techniques. In *Interspeech*. ISCA, August 2011.