Recurrent neural network regularization

Wojciech Zaremba WOJ.ZAREMBA@GMAIL.COM
Google & New York University
Ilya Sutskever ILYASU@GOOGLE.COM
Google
Oriol Vinyals VINYALS@GOOGLE.COM
Google

Abstract

We present a simple regularization technique of recurrent neural networks (RNNs) with long short term memory (LSTM) units. This technique is based on dropout and gives tremendous performance boost. We show that it is beneficial in variety sequence modelling problems like language modeling, speech recognition, and machine translation.

1. Introduction

RNNs yields the state-of-the-art performance on many sequence modelling tasks like language modelling, and speech recognition. Moreover, recent results in machine translation (Cho et al., 2014) shows their potential use in this field as well. However, up today there was no good techniques to regularize them. Various attempts to inject noise or mask some of activations (dropout) were giving just a small improvement (Graves, 2013). This work presents how to augment LSTMs with dropout. Resulting models give a tremendous improvement over the baseline.

2. Related work

There have been extensive work on using RNNs for sequence modelling (Mikolov, 2012; Sutskever, 2013). Moreover, there were considered various twists into common architecture, which potentially can capture long term dependencies (Hochreiter & Schmidhuber, 1997; Graves et al., 2009; Cho et al., 2014). This work choses LSTM architecture, however it is likely that conclusions presented here would extend to other models.

We focus here on tasks, which were previously considered in RNN literature like (1) language modelling, (2) speech recognition, and (3) machine translation. Language modelling is a classical task tackled by RNNs (Pascanu et al., 2013; Mikolov et al., 2010; 2011). Using RNNs for speech recognition was previously considered by (Robinson et al., 1996; Graves et al., 2013). Finally, machine translation is a novel task for which RNNs are used. Usually, they are used for language modelling, re-ranking, or phrase modelling (Cho et al., 2014).

3. Regularized RNN with LSTMs

We describe in this section how LSTMs work 3.1. Next, we show how to regularize them 3.2, and we give some intuitions why it works.

We use upper indexing for layer number, lower indexing for a time step. Our states are n dimensional. $h_k^l \in \mathbb{R}^n$ is a hidden state in layer l in step k. Moreover, for simplicity we denote by $T : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^n$ a linear transform with bias $(Wx+b)$. $\odot$ is a element-wise multiplication. h_k^0 is a input word-vector, and h_k^L is used to predict y_k (L is a number of layers).

3.1. Long-short term memory units

Dynamics of RNNs can be described in terms of transitions from previous hidden states to the hidden states in the next step. That is the function

$$\text{RNN} : h_k^{l-1}, h_{k-1}^l \rightarrow h_k^l$$

In case of classical RNNs this function is described as:

$$h_k^l = f(Th_k^{l-1} + Lh_{k-1}^l)$$

LSTM has more complicated dynamics in order to be able to "memorize" information for extended number of time steps. Memory is stored in cell ($c_k^l \in \mathbb{R}^n$) units. LSTM

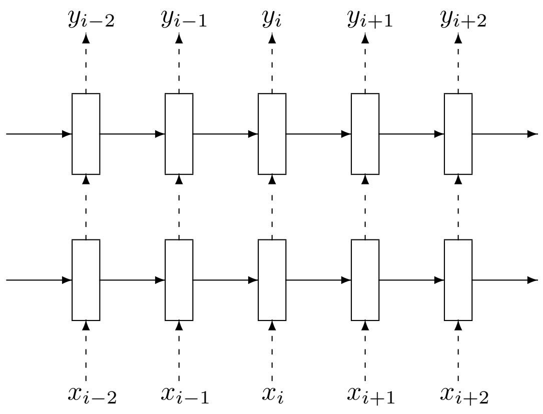

Figure 1. Regularized multilayer RNN. Dashed arrows indicate connections with applied dropout, while solid lines indicate connections where dropout is not applied.

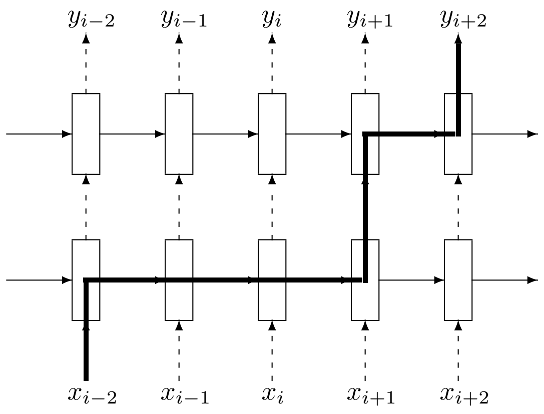

Figure 2. Thick line indicates an exemplary information flow in RNN. Information flow line is crossed $L + 1$ times, where L is depth of network.

performs following mapping.

$$\text{LSTM} : h_k^{l-1}, h_{k-1}^l, c_{k-1}^l \rightarrow h_k^l, c_k^l$$

This mapping is described by:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T \begin{pmatrix} h_k^{l-1} \\ h_{k-1}^l \end{pmatrix}$$
$$c_k^l = f \odot c_{k-1}^l + i \odot g$$
$$h_k^l = o \odot \tanh(c_k^l)$$

Where $\sigma, \tanh$ are applied element-wise.

3.2. Regularization with dropout

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T \begin{pmatrix} \mathbf{D}(h_k^{l-1}) \\ h_{k-1}^l \end{pmatrix}$$
$$c_k^l = f \odot c_{k-1}^l + i \odot g$$
$$h_k^l = o \odot \tanh(c_k^l)$$

Dropout removes part of information, and units have to become more robust while performing intermediate input-intermediate output mapping. Simultaneously we don't want to erase entire information. Especially we would like to facilitate memory about event that happened long time ago. Figure 2 shows a flow of an exemplary information from the event x_{i-2} to the prediction in the step $i + 2$. We can see that information is influenced with dropout only $L + 1$ times, and it is independent of how far in past event occurred. All the previous regularization techniques were constraining recurrent connections, which effectively exponentially fast "blurred" information about the past.

4. Experiments

We present here results in various domains (1) language modeling 4.1, (2) speech recognition 4.2, and (3) machine translation 4.3.

4.1. Language modeling

We have conducted word-level prediction experiments on Penn tree bank (PTB) dataset. This dataset consists of 929k training examples, 73k validation examples, and 82k test examples. It has 10k words in vocabulary.

Our model is a two layer LSTM network, with 650 units initialized uniformly in $[-0.05, 0.05]$. We apply 50% of dropout on non-recurrent connections. We train for 39 epochs, starting with learning rate 1, and after 6 epochs we decrease it by 1.2 in every epoch. We unroll RNN for 35 steps, and clip gradients at 5. We set mini-batch to 20. Table 1 shows our results.

Model	Validation set	Test set
A single model		
Previous state-of-the-art [1]		107.5
Regularized LSTM	**86.2**	**82.7**
Model averaging		
Previous state-of-the-art [2]	83.5[3]	89.4
2 regularized LSTMs	80.6	77.0
5 regularized LSTMs		
10 regularized LSTMs		

Table 1. Word-level perplexity on Penn-tree-bank dataset.

4.2. Speech recognition

4.3. Machine translation

5. Discussion

References

Cho, Kyunghyun, van Merrienboer, Bart, Gulcehre, Caglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Graves, Alex. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Graves, Alex, Liwicki, Marcus, Fernández, Santiago, Bertolami, Roman, Bunke, Horst, and Schmidhuber, Jürgen. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

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

Mikolov, Tomáš. *Statistical language models based on neural networks*. PhD thesis, Ph. D. thesis, Brno University of Technology, 2012.

Mikolov, Tomas, Karafiát, Martin, Burget, Lukas, Cernockỳ, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *INTERSPEECH*, pp. 1045–1048, 2010.

Mikolov, Tomas, Deoras, Anoop, Povey, Daniel, Burget, Lukas, and Cernocky, Jan. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pp. 196–201. IEEE, 2011.

Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Robinson, Tony, Hochberg, Mike, and Renals, Steve. The use of recurrent neural networks in continuous speech recognition. In *Automatic speech and speaker recognition*, pp. 233–258. Springer, 1996.

Sutskever, Ilya. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, 2013.

[1](Pascanu et al., 2013)

[2](Mikolov, 2012)

[3]Weight of individual models are tuned to minimize this score. This few parameters are fit on this validation set, which is not completely fair.