Bilinear recurrent neural networks (draft)

Povilas Daniušis *

April 14, 2017

1 Introduction

Many sequence learning problems are efficiently solved by recurrent neural networks (RNN's) (e.g. speech recognition [15], machine translation [18], image caption generation [21]). RNN's encode a sequence of inputs into the fixed structure internal state, which is further used to infer a sequence of corresponding outputs.

Although from theoretical point of view RNN's are capable of performing arbitrary computations [17], in practice training of RNN's is often non-trivial and time consuming process, and therefore new, more efficient architectures and training methods are needed.

This study is devoted to the question of how RNN's can benefit from bilinear products.

Although the inner product representation plays fundamental role in many machine learning algorithms, they often can benefit from bilinear products as well. For example, bilinear hashing [9], feed forward neural nets [6], SVM's [3] can be more efficient than the conventional analogues.

This article contributes to RNN's by applying bilinear products to derive new modification of long short-term memory (LSTM) [12] and gated recurrent unit (GRU) [5] recurrent neural networks, and conducting an empirical analysis of suggested models. LSTM and GRU were chosen because of their effectiveness.

Main advantages of our approach are that it can be used directly for matrix-valued sequences, is able to capture the structure of such a data more efficiently comparing to conventional analogues, and contains less parameters. Tensorflow [19] implementation of suggested bilinear RNN's can be downloaded from `https://github.com/povidanius/bilinear_rnn`.

*povilas.daniusis@gmail.com

1.1 Short review of RNN models

Let t be discrete time variable, and let $x_t \in \mathbb{R}^{D_x}$ be corresponding input vectors. Simple RNN (SRNN) model is defined by recurrence $h_t = \phi(Wx_t + Uh_{t-1} + b)$, where h_t is hidden state vector, W and U are parameter matrices, b is bias vector, and ϕ - activation function [7]. Receiving the sequence of inputs x_t, the model maintains hidden state h_t, and outputs $y_t = \psi(Vh_t + c)$, where V and c are another parameters, and ψ is output activation function. Although SRNN is able to learn non-trivial sequential regularities, in practice it is rather limited.

1.1.1 LSTM

LSTM [12] relies on the combination of two innovations: gate mechanism and additive state update mechanism.

Upon receiving new input x_t the LSTM combines it with hidden state into new candidate memory vector.

The input gate controls an integration of this new information into cell's memory, allowing to inhibit certain components of candidate cell. Similarly, the forget gate controls integration or previous memory, and output gate determines exposition weights of c_t.

An update of LSTM state $(c_t, h_t)^T$ is defined by:

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\ f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\ o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\ \tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1} + b^c)^{,} \\ c_t &= f_t \bullet c_{t-1} + i_t \bullet \tilde{c}_t \\ h_t &= o_t \bullet \tanh(c_t), \end{aligned} \tag{1}$$

Where $\bullet$ denotes element-wise multiplication, i_t, f_t, o_t are called input, forget and output gates, $\tilde{c}_t$ - candidate cell memory, c_t - cell memory, and h_t - hidden state. All aforementioned variables are D_h- dimensional. Parameter count of LSTM is $4 \cdot D_h \cdot (D_x + D_h + 1)$, and state is defined by $2 \cdot D_h$ variables.

The intuition behind LSTM c_t may be interpreted as internal memory of LSTM, while h_t - represent its content, exposed by the output gate.

1.1.2 GRU

Similar and simpler recurrent architecture, gated recurrent unit (GRU) [5], is defined by:

$$
\begin{aligned}
u_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
r_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
\tilde{h}_t &= th(W^h x_t + r_t \bullet (U^h h_{t-1}) + b^h) \\
h_t &= u_t \bullet \tilde{h}_t + (1 - u_t) \bullet h_{t-1},
\end{aligned}
\tag{2}
$$

Reset gate r_t controls integration of previous state h_{t-1} into candidate state $\tilde{h}_t$, and update gate u_t controls integration of candidate state into state update. GRU has $3 \cdot D_h \cdot (D_x + D_h + 1)$ parameters and D_h-dimensional state variable.

1.2 Bilinear analogues of LSTM and GRU

This section describes bilinear LSTM and GRU RNN's. We assume that the inputs X_t are $D_X^1 \times D_X^2$ matrices, and hidden state variables are $D_H^1 \times D_H^2$ matrices. Bilinear SRNN can now be reformulated as: $H_t = \phi(W_1 X_t W_2 + U_1 H_{t-1} U_2 + B)$, where ϕ is non-linear activation function, and parameter matrices of appropriate dimensions.

We expect that such an approach allows to capture linear row and column structures of the data, and is more compact in terms of parameters (see Table 1).

1.2.1 Bilinear LSTM (BLSTM)

$$
\begin{aligned}
I_t &= \sigma(W_1^i X_t W_2^i + U_1^i H_{t-1} U_2^i + B^i) \\
F_t &= \sigma(W_1^f X_t W_2^f + U_1^f H_{t-1} U_2^f + B^f) \\
O_t &= \sigma(W_1^o X_t W_2^o + U_o^i H_{t-1} U_2^o + B^o) \\
\tilde{C}_t &= \tanh(W_1^c X_t W_2^c + U_1^c H_{t-1} U_2^c + B^c) \\
C_t &= F_t \bullet C_{t-1} + I_t \bullet \tilde{C}_t \\
H_t &= O_t \bullet \tanh(C_t),
\end{aligned}
\tag{3}
$$

1.2.2 Bilinear GRU (BGRU)

$$
\begin{aligned}
R_t &= \sigma(W_1^i X_t W_2^i + U_1^i H_{t-1} U_2^i + B^i) \\
U_t &= \sigma(W_1^f X_t W_2^f + U_1^f H_{t-1} U_2^f + B^f) \\
\tilde{H}_t &= th(W_1^c X_t W_2^c + R_T \bullet (U_1^c H_{t-1} U_2^c) + B^c) \\
H_t &= U_t \bullet \tilde{H}_t + (11^T - U_t) \bullet H_{t-1},
\end{aligned}
$$

Method	Parameter dimension	State dimension
LSTM	$4 \cdot D_h \cdot (D_x + D_h + 1)$	$2D_h$
BLSTM	$4(D_H^1(D_X^1 + D_H^1 + D_H^2) + D_H^2(D_X^2 + D_H^2))$	$2D_H^1 D_H^2$
GRU	$3 \cdot D_h \cdot (D_x + D_h + 1)$	D_h
BGRU	$3(D_H^1(D_X^1 + D_H^1 + D_H^2) + D_H^2(D_X^2 + D_H^2))$	$D_H^1 D_H^2$

Table 1: Parameter and state space dimensionalities

2 Computer experiments

This section is devoted to computer experiments of suggested bilinear RNN's.

2.1 Meta-learning of regularizer

2.2 Meta-learning of RNN

In this experiment we follow scheme described in [1] and try to learn parameters of conventional LSTM using BLSTM.

References

[1] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas N. Learning to learn by gradient descent by gradient descent. CoRR, abs/1606.04474, 2016.

[2] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on, 5(2):157-166, 1994.

[3] Cai, D., He, X., Han, J. Learning with Tensor Representation, preprint, 2006.

[4] Cheng, Y., Yu, F. X., Feris, R.,S., Kumar, S., Choudhary, A., and Chang, S. An exploration of parameter redundancy in deep networks with circulant projections. In ICCV, 2015.

[5] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS Deep Learning Workshop, 2014.

[6] Daniusis, P., and Vaitkus, Pr. Neural network with matrix inputs. Informatica, 2008, vol.19 (4): 477-486.

[7] Elman, J. L. Finding structure in time. CRL Technical Report 8801, Center for Research in Language, University of California, San Diego, 1988.

[8] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwi nska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. Nature, 2016.

[9] Gong, Y., Kumar, S., Rowley, H. A., and Lazebnik, S. Learning binary codes for high-dimensional data using bilinear projections. CVPR, pages 484-491, 2013.

[10] Haykin, S. Neural Networks: A Comprehensive Foundation. 2nd Edition. Prentice Hall, 1998.

[11] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. Exploiting the Circulant Structure of Tracking-by-Detection with Kernels. In ECCV, 2012.

[12] Hochreiter, S. and Schmidhuber J. Long Short-Term Memory. Neural Computation, 9(8): pp. 1735-1780, 1997.

[13] Oppenheim, A. V., Schafer, R. W., Buck, J. R. Discrete-time signal processing, volume 5. Prentice Hall Upper Saddle River, 1999.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vander-plas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[15] Sak, H., Senior, A., Rao, K. and Beaufays, F. Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition. In INTERSPEECH, 2015.

[16] Schmidhuber, J. Deep learning in neural networks: An overview. Neural Networks, 61: 85-117, 2015.

[17] Siegelmann, H. T., and Sontag, E. D. On the computational power of neural nets. Journal of Computer and System Sciences, 50, 1995.

[18] Sutskever, I., Vinyals, O., and Le, Q. V.. Sequence to sequence learning with neural networks, pp. NIPS 2014.

[19] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G., Davis A., Dean J., Devin M., et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016. arXiv:1603.04467

[20] Tieleman, T., and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4, 2012.

[21] Vinyals, O., Bengio, S., Erhan, D. Show and tell: A neural image caption generator. In CVPR, 2015.

[22] Yu, F., Kumar, S., Gong, Y, and Chang, S.,-F. Circulant binary embedding. In ICML, Beijing, China, 2014, pp. 946-954.

[23] Yu, F. X., Kumar, S., Rowley, H., and Chang, S.-F. Compact nonlinear maps and circulant extensions. ArXiv preprint arXiv:1503.03893, 2015.

[24] Zhang, M., McCarthy, Z., Finn, C., Levine, S. and Abbeel, P. Learning deep neural network policies with continuous memory states, in IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 520-527.