Bilinear recurrent neural networks (draft)

Povilas Daniušis [*]

May 13, 2017

1 Introduction

Many sequence learning problems are efficiently solved by recurrent neural networks (RNN's) (e.g. speech recognition [17], machine translation [20], image caption generation [23]). RNN's encode a sequence of inputs into state variable of fixed structure, which is further used to infer a sequence of corresponding outputs.

Although theoretically RNN's are capable of performing arbitrary computations [19], 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.

Motivated by successful applications of bilinear projections in various machine learning methods (e.g. bilinear hashing [9], feed forward neural nets [6], SVM's [3]), we devote this study to an analysis of their effectiveness in RNN's.

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 practical effectiveness.

Main advantages of our approach are that it can be used directly for matrix-valued sequences, is able to exploit the structure of such a data more efficiently comparing to conventional analogues, and contains less parameters. Tensorflow [21] implementation of suggested bilinear RNN's can be downloaded from `https://github.com/povidanius/bilinear_rnn`.

1.1 Short review of RNN models

In this section we review Elman (simple RNN), GRU and LSTM models, since they will be useful for our research. We refer the reader to [10], and [18]

[*]povilas.daniusis@gmail.com

for more comprehensive reviews of various neural network models (including RNN).

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

$$h_t = \phi(Wx_t + Uh_{t-1} + b), \tag{1}$$

where h_t is 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 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. The limitations of SRNN arise from so called gradient vanishing/exploding effect [2].

1.1.1 Long short-term memory

Long short-term memory (LSTM) [12] deals with aforementioned drawbacks of SRNN by refining the state update scheme. It relies on the combination of two innovations: gate mechanism and additive state updates. The state variable of LSTM is a pair of vectors (c_t, h_t), which may be interpreted as "long" and "short" type memories.

After receiving new input datum x_t the LSTM combines it with h_{t-1} into new candidate memory $\tilde{c}_t$, and gate variables i_t, f_t and o_t.

The input gate i_t controls the integration of candidate $\tilde{c}_t$ into c_t, allowing to inhibit or activate certain components of candidate cell. Similarly, the forget gate f_t controls integration or previous memory c_{t-1}, and output gate o_t multiplicatively exposes c_t into new hidden state h_t.

Formally, LSTM model is defined by:

$$\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\
\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1} + b^c)' \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t),
\end{aligned} \tag{2}$$

where $\odot$ denotes element-wise multiplication, σ and tanh denotes sigmoid and hyperbolic tangent functions. All the gates and state variables are D_h - dimensional vectors. LSTM is defined by $4 \cdot D_h \cdot (D_x + D_h + 1)$, and state is defined by $2 \cdot D_h$ variables.

1.1.2 Gated recurrent unit

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

$$
\begin{aligned}
u_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
r_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
\tilde{h}_t &= \tanh(W^h x_t + r_t \odot (U^h h_{t-1}) + b^h) \\
h_t &= u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1},
\end{aligned} \tag{3}
$$

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. The effectiveness of GRU and LSTM is often very similar [13].

2 RNN with bilinear projections

Linear projections plays fundamental role in many machine learning algorithms. However, in its conventional form it ignores the internal structure of the data, which may be important.

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:

$$
H_t = \phi(W_1 X_t W_2 + U_1 H_{t-1} U_2 + B), \tag{4}
$$

where ϕ is non-linear activation function, and parameter matrices W_1,W_2,U_1, U_2 and B of appropriate dimensions.

We hypothesise that bilinear projections allow to exploit linear row-column structures of input matrix. Comparing to conventional RNN's, bilinear RNN's are also more compact in terms of parameters (see Table 1).

2.0.3 Bilinear LSTM (BLSTM)

$$
\begin{aligned}
I_t &= \sigma(W_1^i X_t W_2^i + U_1^i H_{t-1} U_2^i + B^i) \\
F_t &= \sigma(W_1^f X_t W_2^f + U_1^f H_{t-1} U_2^f + B^f) \\
O_t &= \sigma(W_1^o X_t W_2^o + U_o^i H_{t-1} U_2^o + B^o) \\
\tilde{C}_t &= \tanh(W_1^c X_t W_2^c + U_1^c H_{t-1} U_2^c + B^c) \\
C_t &= F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \\
H_t &= O_t \odot \tanh(C_t),
\end{aligned}. \tag{5}
$$

RNN model	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.

2.0.4 Bilinear GRU (BGRU)

$$\begin{aligned} R_t &= \sigma(W_1^i X_t W_2^i + U_1^i H_{t-1} U_2^i + B^i) \\ U_t &= \sigma(W_1^f X_t W_2^f + U_1^f H_{t-1} U_2^f + B^f) \\ \tilde{H}_t &= \tanh(W_1^c X_t W_2^c + R_T \odot (U_1^c H_{t-1} U_2^c) + B^c) \\ H_t &= U_t \odot \tilde{H}_t + (1_{D_H^1} 1_{D_H^2}^T - U_t) \odot H_{t-1}, \end{aligned} \tag{6}$$

where $1_d \in \mathbb{R}^d$ is column vector of ones. All the variables in Eq. (5) and Eq. (6) are $D_H^1 \times D_H^2$ dimensional matrices.

3 Computer experiments

In this section we describe computer experiments with suggested bilinear RNN's.

3.1 Patterns in a sequence of matrices

We will investigate detection of patterns in a sequence of input matrices. We generate toy data set consisting of 2000 training, and 1000 testing sequences of matrix inputs and matrix outputs. Each input sequence consists of 20 16×16 matrices $X_1, X_2, ...X_{20}$ composed of i.i.d. Gaussian entries, and 16×16 output matrix defined by so that it would depend both from long and short range $Y = \max(X_2, \frac{1}{2}(X_{19}^T + X_{20}))$. We estimate the output $Y = UH_{20}V + B$, where U, V and B are parameter matrices of appropriate dimensions. We train RNN models for 1000 epochs using Adam [14] optimizer, and data batches of 128 elements. In Table 2 we report mean squared errors on the testing set.

4 Conclusions

We applied bilinear products to derive modifications of popular recurrent neural networks: Elman (SRNN), GRU and LSTM. Resulting models are well suited for matrix-valued data sequences, are more economical in terms of

RNN model	Performance	Parameter count
LSTM	0.0	0.0
BLSTM	0.0	0.0
GRU	0.0	0.0
BGRU	0.0	0.0
SRNN	0.0	0.0
BSRNN	0.0	0.0

Table 2: Performance comparison (detection of patterns in matrix sequences).

parameters. Although standard RNN's are also applicable to matrix-valued inputs of specific dimensions ($D_X^2 = D_H^2$), they omit linear projections of columns, which may be important. Computer experiments also (hopefully will) confirm our hypothesis, that bilinear product allow to exploit structure of data matrices in recurrent neural networks.

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] Daniušis, 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] Jozefowicz, R., Zaremba, W., and Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In ICML2015, p.p. 2342-2350, 2015.

[14] Kingma, D., and Ba, J. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG], December 2014.

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

[16] 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.

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

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

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

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

[21] 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

[22] 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.

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

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

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

[26] 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.