Sequence modeling using a memory controller extension for LSTM

Itamar Ben-Ari
Intel, Advanced Analytics
itamar.ben-ari@intel.com

Ravid Shwartz-Ziv
Intel, Advanced Analytics
ravid.ziv@intel.com *

Abstract

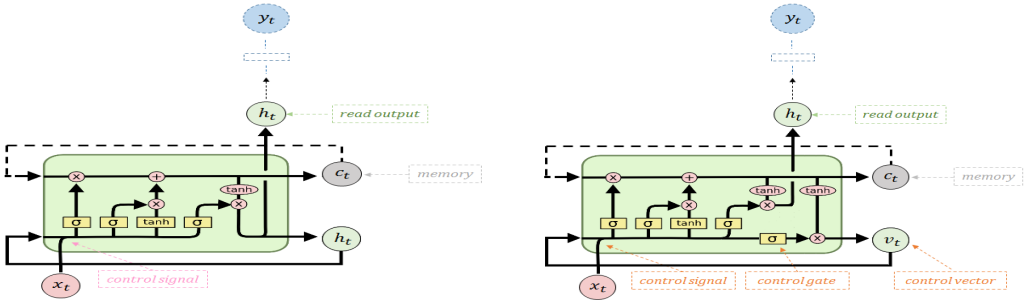
The Long Short-term Memory (LSTM) recurrent neural network is a powerful model for time series forecasting and various temporal tasks. In this work we extend the standard LSTM architecture by augmenting it with an additional gate which produces a memory control vector signal inspired by the Differentiable Neural Computer (DNC) model. This vector is fed back to the LSTM instead of the original output prediction. By decoupling the LSTM prediction from its role as a memory controller we allow each output to specialize in its own task. The result is that our LSTM prediction is dependent on its memory state and not the other way around (as in standard LSTM). We demonstrate our architecture on two time-series forecast tasks and show that our model achieves up to 8% lower loss than the standard LSTM model.

1 Introduction

Hochreiter and Schmidhuber [1] introduced the long short term memory (LSTM) as a novel learning architecture to solve hard long time-lag problems. The key idea of that work was the introduction of a memory cell which can maintain its state over long time periods. The cell is augmented with a gating mechanism which controls the information flow through the cell. LSTM networks have been used for various temporal tasks. This includes language modeling [2] machine translation [3], analysis of audio [4] and video [5], acoustic modeling of speech [6] and modeling clinical data [7]. The Differentiable Neural Computer (DNC) model [8] also maintains a memory cell augmented with a rich memory management mechanism which allows it to address memory locations sequentially (in the order they were written), associatively (based on content similarity) and based on usage metrics (used by a memory reallocation mechanism). The DNC is composed of two conceptual units which are analogous to a computer CPU and RAM. The memory addressing mechanism (a.k.a Memory Access Unit) is controlled by a memory controller which signals the MAU what to read and write by forwarding it a control vector. The MAU translates this vector to actual memory addresses. From this perspective the LSTM output, which is fed back to the LSTM (concatenated with the sequence input) can be seen as a memory control vector which is translated to memory addresses through the various LSTM gates. The double role of the LSTM output as a memory controller and predictor hinders the LSTM performance. This paper proposes a novel LSTM architecture which decouples the LSTM output prediction from its memory management role by introducing a new memory control gate. This MC gate produces a second output vector, a memory control vector which is fed back to the LSTM as opposed to the prediction vector which is forwarded as the LSTM output prediction (see 1).

*These two authors contributed equally

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



(a) LSTM (b) MC-LSTM

Figure 1: An illustration of the LSTM and MC-LSTM topologies. In the MC-LSTM topology a new (control) vector v_t is fed back to the network instead of h_t. This vector is a function of the memory cell c_{t-1} and the vector $[v_{t-1}, x_t]$ and is not depended on h_{t-1}.

2 Related work

Since it's original formulation, many improvements have been introduced to the LSTM architecture. [9] introduced "peephole" connections to the LSTM unit that connect the memory cell to the gates. [10] introduced linear projection layers between the LSTM units layer and the output layer in order to reduce the amount of parameters for networks with many LSTM blocks. Their results significantly improved performance in a large vocabulary speech recognition task. [11] introduced the Dynamic Cortex Memory. By adding recurrent connections between the gates of a single block they improved the convergence speed of the LSTM. [12] were able to improve the performance of the LSTM on an off-line handwriting recognition dataset by introducing a trainable scaling parameter for the slope of the gate activation functions. [13] performed a detailed comparison of popular LSTM variants without conclusive results. [2] found that standard LSTM architectures, when properly regularized, outperform more recent state of the art RNN models.

3 The LSTM model

The central idea behind the LSTM architecture is to incorporate a memory cell that can maintain its state over long time periods. The cell is augmented with a non-linear gating mechanism which regulates the information flow through the memory cell. At time step t, the LSTM output from the previous step h_{t-1} is concatenated with the sequence current input x_t to a vector $[h_{t-1}, x_t]$, which is fed back to the LSTM (see 1). This vector controls the LSTM memory through the various gates. Each gate is implemented by a sigmoid function σ composed with a linear function and produces a memory (fuzzy) address in the form of a gray-scale mask which masks the non-relevant memory slots for each operation. The LSTM memory update equations at time t are defined as follows:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{//get erase address} \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{//get write address} \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{//get read address} \\
n_t &= \tanh\left(W_n[h_{t-1}, x_t] + b_n\right) \quad \text{//create new content} \\
c_t &= f_t * c_{t-1} + i_t * n_t \quad \text{//erase old content and write (add) new content} \\
h_t &= o_t * \tanh\left(c_t\right) \quad \text{//read from (updated) memory}
\end{aligned}
$$

4 The MC-LSTM model

In our model we introduce a new 'memory control' gate. This gate produces a memory address m_t which is used to extract the next control vector v_t (it can also be thought of as reading the next 'command' to be executed in analogy to a computer). The original LSTM input vector $[h_{t-1}, x_t]$

controlling the memory is replaced by the vector $[v_{t-1}, x_t]$ in all gates equations. The full set of equations defining the MC-LSTM is as follows:

$$
\begin{aligned}
f_t &= \sigma(W_f[v_{t-1}, x_t] + b_f) \quad \text{//get erase address} \\
i_t &= \sigma(W_i[v_{t-1}, x_t] + b_i) \quad \text{//get write address} \\
o_t &= \sigma(W_o[v_{t-1}, x_t] + b_o) \quad \text{//get read address} \\
m_t &= \sigma(W_m[v_{t-1}, x_t] + b_m) \quad \text{//get control address} \\
n_t &= \tanh(W_n[v_{t-1}, x_t] + b_n) \quad \text{//create new content} \\
c_t &= f_t * c_{t-1} + i_t * n_t \quad \text{//erase old content and write (add) new content} \\
h_t &= o_t * \tanh(c_t) \quad \text{//read prediction} \\
v_t &= m_t * \tanh(c_t) \quad \text{//read next command}
\end{aligned}
$$

The memory update equations are no longer dependent on the output prediction h_{t-1}. This allows it to be tuned separately to give the correct output. All gates equations are now functions of the MC vector v_{t-1} (and x_t) which is trained to optimize memory operations outcome.

5 Experiments

In our experiments we tried to predict the joint probability of a sequence suffix given its prefix. For an input sequence $(x_0, ...x_N) \in R^{l \times N}$ where l is the signal step dimension and for a prefix size m we predict the conditional density $p(x_{m+1}, ..., x_N | x_0, ..., x_m) = \Pi_{i=0}^{N-m-1} p(x_{m+i+1} | x_0, ..., x_{m+i})$ where we assume $x_{m+i+1} | x_0, ..., x_{m+i} \sim \mathcal{N}(\mu_{m+i+1}, \sigma_{m+i+1})$ for mean vector μ_{m+i+1} and variance matrix diagonal σ_{m+i+1} (in our experiments we tested uncorrelated matrices predictions only). An extension to the Gaussian mixture case can be found in [14]. We added two fully connected layers to the LSTM prediction output h_t at time t: $\tanh(W_\mu h_t + b_\mu) = \mu_{t+1}$ and $\sigma(W_\sigma h_t + b_\sigma) = \sigma_{t+1}$ which output the conditional density parameters for x_{t+1}. We used the log likelihood loss in the training phase $-\log \Pi_{i=0}^{N-m-1} p(x_{m+i+1} | x_0, ..., x_{m+i}, \mu_{m+i+1}, \sigma_{m+i+1})$. This loss can also be used as an indicator for an anomaly in the sequence when low likelihood values occur.

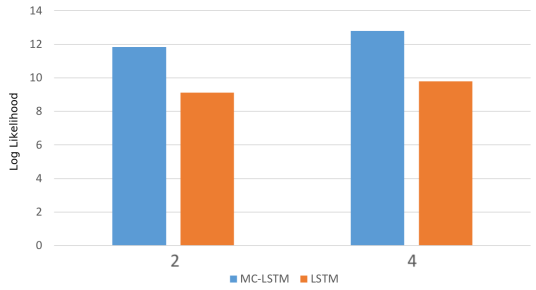
5.1 Sum of sines signal

In this experiment we tried to model randomly generated signals composed of sum of sines. Each generated signal was 200 steps long. We used a prefix size of 20 steps and signal step dimension $l = 3$. Each signal was composed of a sum of randomly generated sines where the phase and amplitude were randomly generated. The frequency of the first sine in the sum was also randomly generated but the frequencies of the other sines were set to be multiplications of the first sine's frequency in order to assure a signal repetition inside the 200 steps window. We normalized the signals in each dimension such that their values fell inside the interval $[-1, 1]$. We evaluated our model on signals with different number of sines components. By increasing the number of sines components we were able to increase the complexity of the signal and measure how well each model handle it (see 2 and 3). We also challenged the models by reducing their memory cell size (see 3b). In both cases our model outperformed the standard LSTM.

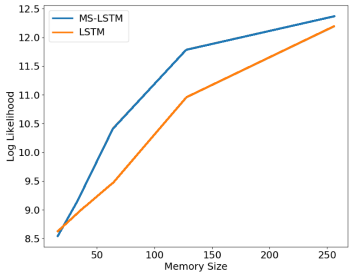
(a) 2 sines component signal

(b) 4 sines component signal

Figure 2: Two examples of randomly generated signals with different number of sine components, step dimension $l = 3$ and window size of 200.



(a) Different number of sines

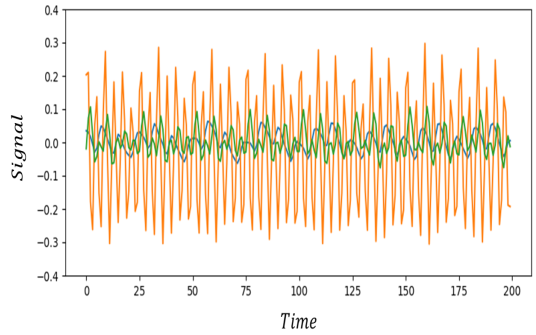


(b) Different number of memory sizes

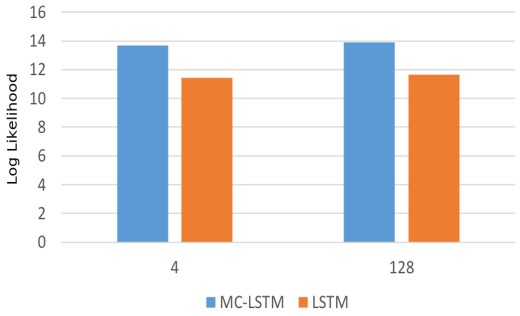
Figure 3: Comparison between LSTM and MC-LSTM for different memory cell sizes and number of sines components in the signal. In figure (a) the memory size was set to 8. In figure (b) the number of sine components was set to 4.

5.2 Fan Filter Unit dataset

In this experiment we tested our model on an internal data set containing an accelerometer sensor signal sampled at 1kHz which was coming from a fan filter unit placed in one of our factories. The sensor was used for the purpose of monitoring and anomaly detection (see 4a). In figure 4b we show the log likelihood of the models on two different memory cell size configurations. In both cases our model outperformed the standard LSTM.



(a) Accelerometer signal



(b) Performance comparison between the LSTM model and our MC-LSTM for different memory sizes

Figure 4: Accelerometer dataset

6 CONCLUSIONS

We proposed a memory controlled LSTM model which decouples the LSTM prediction output from its roll as a memory controller. This enables the LSTM output prediction to be dependent on the LSTM state (memory) and not the other way around as in standard LSTM. It allows the LSTM more flexibility in its prediction and memory management. We show that our model outperforms the standard LSTM on two time-series forecast tasks given different configurations of the problem and model memory cell size.

References

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

[2] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589, 2017.

[3] Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*, 2017.

[4] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertilä, Toni Heittola, and Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. *CoRR*, abs/1706.02293, 2017.

[5] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[6] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[7] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to diagnose with LSTM recurrent neural networks. *CoRR*, abs/1511.03677, 2015.

[8] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[9] F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194 vol.3. IEEE, 2000.

[10] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014.

[11] Sebastian Otte, Marcus Liwicki, and Andreas Zell. *Dynamic Cortex Memory: Enhancing Recurrent Neural Networks for Gradient-Based Sequence Learning*, pages 1–8. Springer International Publishing, Cham, 2014.

[12] Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *International Conference on Frontiers in Handwriting Recognition*, pages 279–284, Crete, Greece, September 2014.

[13] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015.

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