Clockwork RNN

Paul Mustière - David Panou

10 décembre 2016

1 Abstract

Recurrent Neural Networks have been successfully proved to able to predict sequence outputs for a given input sequence. Such abilities have turned them into being used in a variety of tasks ranging from Text translation. However, training long-term memories with recurrent neural networks have been proven to be a difficult tasks because of Vanishing and Exploding Gradient issues. Such issues have led to the development of LSTM. This paper introduces a novel way of splitting the the shared weight parameters in order to specialize each cells on different time-related dependencies leading to a better performance of the simple RNN architecture both in training and generation.

2 Context

2.1 Recurrent Neural Networks

A RNN [3] is a simple extension of the feedforward neural networks for processing sequential data.

More formally, given a sequence $x = (x_1, x_2, ..., x_T)$ the RNN updates its recurrent hidden state h_t by

Where ϕ is a nonlinear function that is usually the composition of the cell activation and an activation function.

The update function of the recurrent hidden state is usually implemented has :

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

Where g is a smooth bounded function such as a sigmoid function of *thanh*, W the input weights matrix and U the hidden weight matrix.

The sequence probability can be decomposed using the chain-rule into

$$p(x_1, x_2, ..., x_T) = p(x_1)p(x_2|x_1)...p(x_T|x_1, ...x_{T-1})$$

Those models are known as generative RNN or simple RNN (SRN).

$$\mathbf{h}_t = \begin{cases} 0, & \text{if } t = 0 \\ \phi(\mathbf{h}_t - 1, \mathbf{x}_t), & \text{otherwise} \end{cases} \tag{1}$$

Hidden unit update in SRN

Despite the recents success of RNN applications in translation and speech, vanilla RNN are never used in the those tasks. This issue is due to their lack of ability to capture long dependency trend, more formally stated by the Vanishing Gradient or Exploding Gradient issues [1].

Those issue, makes the Gradient Based optimization methods struggle because of the effect of the short-term dependencies.

This issue has led to further attempts to redefine recurrent neural networks with more sophisticated activation functions in order to adapt them to support longer input dependencies. The well established (Long Short Term Memory) LSTM units [2] are now known to perform well at capturing long-term dependencies and are used in many recent tasks such as machine translation [?]. More recents work to adapt the SRN to support longer time lag during backpropagation include the Gated Recurrent Networks, that control the flow of the back propagated error [5].

Despite the interesting approach such methods, by totally exposing the full content of the cell without any control, GRU are diverging from the LSTM approach, a approach that clockwork-RNN refines, we won't go into further detail in this type of approaches. Interested reader can find a comparative studies of both methods in [6].

Other approches focuses on adapting the Gradient Descent algorithms. Such attempts includes Hessian Free optimization [7]. However, those methods can be coupled with LSTM, GRU or CW-RNN so we won't go into further details.

2.2 LSTM

As specified earlier, LSTM uses a specialized architecture that allows information to be stored in a linear unit called *Constant Error Carousel* indefinitely.

Unlike to the recurrent unit which simply computes a weighted sum of the input signal and applies a nonlinear function, each j-th LSTM unit maintains a memory c_t^j at time t. An activation is h_t^j is computed based on the content of a memory cell c_j^t and an *output gate* that controls the amount of memory exposed to the current information o_t^j. The existing memory is forgotten with respect to the value of a *forget gate* and the new memory content is added following the value of a input gate i_t^j.

The following equations describe the comportement of those gates and cells

$$\begin{aligned}
h_t^j &= o_t^j tanh(c_t^j) \\
o_t^j &= \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_t + V_o \mathbf{c}_t) \\
c_t^j &= f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \\
\tilde{c}_t^j &= tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})^j
\end{aligned}$$

With V, U and W respectively the weight matrices related to the memory cell, the hidden units and the input units. It is clear that unlike the traditional recurrent units that overrides its content at each time-step, an LSTM units can decide wether to keep its memory thanks to its gate.

3 Clockwork RNN

CW-RNN propose a novel RNN Architecture in which the hidden layer is partitioned into separate modules, running at different time clocks, hence the name.

This partitioning is done via a decomposition of the input weight matrix U and W the output weight matrix into g block size matrices, W_H thus becoming a block upper triangular matrix where each block correspond to a time step weights parameters.

$$\mathbf{U} = \begin{pmatrix} \mathbf{W}_{H_1} \\ . \\ . \\ . \\ \mathbf{W}_{H_g} \end{pmatrix}$$

The update of a specific block weights follow the following scheme :

$$\mathbf{U}_i = \begin{cases} \mathbf{U}_{H_i} & \text{for } t \mod t_i = 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (2)$$

As a result, the low-clock-rate modules process, retain and output the long-term information obtained from the input sequences.

4 CW-RNN training

There are several ways to train recurrent neural networks each with its pros and cons. The most popular methods is a direct adaptation of Back Propagation, called Back Propagation Through Time (BPTT) [8] [9] that is simply the application of the Gradient Descent optimization scheme to the unfolded RNN. The drawback of BPTT can't be parallelized and is hard to be used in an online paradigm. Other implementation of backpropagation methods includes
This paper introduces an adaptation of the BPTT algorithm to fit it to the CW-RNN models.

5 Project roadmap

5.1 Implementation

Our main objective during this project is to replicate the results obtained in the paper especially the music sequence generation task that "only" involves 320 data points in a [-1,1] scaled, and a hundred of parameters.
A useful implementation would be to compare CW-RNN to vanilla RNN and LSTM has the paper does. If we find time, we will extend our implementation to [?] that uses a context layer in order to catch longer dependencies such as topic information such as a cache models [?]

5.2 Reinforcement Learning

The IAR module is actually a Robotic course so we would like to make a connection and adapt this model to use it it this context. Using Recurrent Q-Learning [] is an intuitively approach to use RNN to learn the Q-Values and has been used to learn and retain "history features".
Unfortunately, such methods works well on simple problems and small datasets but have . More recently, Deep Q-Learning [] has yielded proficient controllers for complex tasks. RNN have been successfully stacked with DQN in order to strengthen the Markov Decision Process, leading more robust performances when input are altered.

If we find time, interesting experience would be to test the implementations through the OpenAI gym framework [10] on, the limit being the computational power required.

Références

[1] Hochreiter, Sepp. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02) : 107116, 1998.

[2] Hochreiter, Sepp and Schmidhuber, Ju rgen. Long short-term memory. Neural computation, 9(8) : 17351780, 1997.

[3] Elman, Jeffrey L. Finding structure in time. Cognitive science, 14(2) :179211, 1990.

[4] Sequence to Sequence With Neural Networks (Sutskever et al., Google, NIPS 2014)

[5] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2015a). Gated feedback recurrent neural networks. In ICML15.

[6] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS2014 Deep Learning workshop, arXiv 1412.3555.

[7] Martens, J. and Sutskever, I. Learning recurrent neural net- works with Hessian-free optimization. In Getoor, L. and Scheffer, T. (eds.), Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 1033 1040, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

[8] Kuhn, Roland and De Mori, Renato. A cache-based natural language model for speech recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 12(6) :570583, 1990.

[9] Williams, Ronald J and Zipser, David. Gradient-based learning algorithms for recurrent networks and their computational complexity. Back-propagation : Theory, architectures and applications, pp. 433486, 1995.

[10] Werbos, Paul J. Generalization of backpropagation with application to a recurrent gas market model. Neural Networks, 1(4) :339356, 1988.

[11] https ://gym.openai.com/