Assignment 13

Recurrent Neural Networks for Sequence Learning Review

Nisim Hurst

Sunday 22 April 2018

Abstract

Recurrent neural networks capture long term relationships between correlated inputs in time and space. They have been traditionally hard to train but recent developments in architecture, optimization techniques and parallel computing led these kind of online training to be feasible. The paper [1] makes a review of techniques and applications in which variants of recurrent neural networks have tried-and-true benefits.

Hypothesis

The high expressiveness of cycles in networks nodes will capture the dynamics of sequences. An arbitrary length context memory that doesn't rely on the assumption of independence between samples (i.e. they are related in time or space) will give better results for long term relationships that cannot be properly learned by using other tools such as hidden Markov models.

These relationships are quite expensive to learn using other methodologies, in fact, the optimization is an NP-complete problem. However using recurrent neural networks, recent advances in architecture and parallel computing will cheapen the computational cost of training millions of parameters.

Evidence and Results

The results presented by this paper related with the hypothesis are mainly mathematical constructs that allow each of the architectures excel over a distinct task. First, benefits of simple neural networks are posed, namely: 1. Low complexity, 2. High expressivity and 3. Long term capacity (in contrast with hidden Markov chains). Also, other aspects that improve reaching the global optimum faster are mentioned like the development of ReLU, AdaGrad or RMSProp. Then, starting at section 3, recurrent neural networks are explained in depth. Thus, the following concepts are explained:

1. **Unfolded representation notation.** A representation in which each time step in depicted as a separate network when in fact is the same. Activation function is also called link function.
2. **Back propagation through time.** forward propagation can be translated trivially to RNNs. Just add up a third term comprising the product of the weights between activations and the activation of the previous time step, $W_{aa}a^{t-1}$, inside the activation function $a^{<t>} = g(W_{aa}a^{t-1} + W_{ax}X^{<t>} + ba)$. However, backward propagation is a little bit messier. You have to calculate de loss function for each time step and then add them up to propagate them using the derivative of the activation function at each time step. This process is called backpropagation through time introduced by Werbos on 1990.
3. **Hopfield nets.** RNNs in which the activation function is a simple zero threshold.
4. **Jordan network.** Uses an special unit that are connected to the outputs and then the produced values are feeded back in the network.
5. **Elman network.** RNN that has a single self-connected recurrent edge. Harbinger of the LSTM.
6. **Vanishing and exploding gradients.** it can be solved by one of the following methods:
    1. Regularization penalty over complexity.
    2. Truncated back propagation through time. Some maximum number of time steps at which the error is propagated is set. It sacrifices long term dependencies.

7. **GPU frameworks.** Forward and Back propagation can use GPUs and make the training faster, like Theano or Torch. Also Hessian free methods were developed for GPUs as an alternative to the Newton method that tends to get stuck in saddle points.

Section 4 now focus on the two most important types of RNN's, namely, *Long Short-term Memory* (LSTM) and *Bidirectional Recurrent Neural Networks* (BRNN):

1. **LSTM.** Simple RNNs have long-term memory on their weights. The also have short-term memory in their ephemeral activations. LTSM introduces a kind of intermediate storage via a memory cell. *Constant error carrousel* keep under control the gradients from exploding or vanishing. LSTMs memory cells are composite units that have the following special units/states:
    1. **Input gate.** Block or let pass through the signals from the input nodes.
    2. **Internal state.** A linear activation function coming from a self connected edge.
    3. **Forget gate.** Help to flush the contents of the internal state.
    4. **Output gate.** The output of the memory cell is the value of the output gate multiplied by the value of the internal state.
    5. **Peephole connections.** Connect the internal state to input and output gates without having to be modulated by the output gates.
    6. **tanh activation function.** This function is better than the sigmoid because it has a widder dynamic range.
2. **BRNN.** Two layers of hidden nodes are connected to input and output. On has connections from the past activations and the other from the future activations. This kind of net has the limitation that a fixed endpoint is needed in both the future and the past, thus it is not appropriate for online learning.
3. **Neural Turing Machines.** The authors also briefly mention *Neural Turing Machines* (NTM). This RNNs just have an external addressable memory which improves their performance on algorithmic tasks. It has a controller and memory. The memory has many heads that can write and read concurrently. Also is differentiable from end-to-end so BPTT can be applied. Also experiments show that NTMs generalize better than LTSMs.

Even though the paper is mainly focused on the theoretical support for each architecture over the tasks, section 5 focus on the applications of LSTMs and BRNNs in the following areas of natural language processing:

1. **Data representations for natural language.** Some methods mentioned are: Word embedding's like Word2vec and Glove, recursive autoencodes (RAE).
2. **Current metrics and their evolution over time.** The BLEU score is the geometric mean of the n-gram precisions until reaching a upper limit of N. It is common to both captioning and translation. However, this metric is not correlated to human scores on the long term. METEOR is other metric that overcomes the weaknesses of BLEU by exploring stemming and synonyms.
3. **Language Translation.** Bag of words loose order relationships. The paper cites 2 papers about LSTM usage along with their BLEU scores and details of implementation.
4. **Image Captioning.** This is the process of generating text from an image. Several strategies are compared from about 7 papers.
5. **Further applications.** further applications include handwriting recognition, video captioning, and an adding algorithm that makes a single left to right pass with an accuracy of 99%.

Contributions

In general, the authors present a comprehensive review of methods that first made useful and then made practical this well known models and their intuitions.

The first concrete contribution of the paper is the unified notation by which the authors refer for explaining each of the architectures. All the images of the architectures and their inner workings are unified under this standard or explained thoroughly.

A second contribution is the chronological review of architectures/methods and their evolution until reaching the LSTM and BRNN networks. The authors put together their characteristics along with their benefits and

inconveniences.

Finally, in section 5 the authors cite several applied architectures on 5 different tasks, first explaining out the methods used to measure accuracy on such esoteric and convoluted networks.

Weaknesses

The paper only cites the results of other papers over a narrow task scope. It would be desirable that the authors would assembly a comprehensive tabulated benchmark of all the literature instead of just citing the articles, sometimes even obviating from the paper the empirical results obtained back then.

Another weakness is the extensive introduction of simple neural networks. Densely connected neural networks are related but the relationship is not so strong to justify a complete review of those networks.

Future Work

In the discussion section, the authors say that the body of research that has been started by this article can be used as a solution space for genetic algorithms to find the best architectures appropriate to each task type. However, future work in this area could be to characterize the problems by first selecting a set of features that could be relevant. Those set of features by kind of architecture are still esoteric and ignored by almost any of the nowadays papers.

References

[1] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," pp. 1–38, 2015.