SEQUENCE MODELING WITH RECURRENT TENSOR NETWORKS

Richard Kelley*
Eigenmancy, LLC
Reno, NV 89501, USA
richard@eigenmancy.com

ABSTRACT

We introduce the recurrent tensor network, a recurrent neural network model that replaces the matrix-vector multiplications of a standard recurrent neural network with bilinear tensor products. We compare its performance against networks that employ long short-term memory (LSTM) networks. Our results demonstrate that using tensors to capture the interactions between network inputs and history can lead to substantial improvement in predictive performance on the language modeling task.

1 INTRODUCTION

Sequence modeling is a fundamental task in many domains of interest in artificial intelligence, ranging from machine translation and speech recognition to robot state estimation and localization. Recently, a number of sequence modeling problems have been successfully approached by way of recurrent neural networks (RNNs), which achieve state-of-the-art results on a wide array of tasks. Recurrent neural networks allow directed cycles in the connections between units, which commonly feed into *hidden state units* that implement a form of "memory" for the network.

Although there are a number of variations on the basic concept of recurrent neural networks, one feature all current models share is that interactions between input vectors and hidden state vectors take place indirectly, through addition and some kind of nonlinear transformation. In this paper, we introduce the recurrent tensor network, which uses a bilinear tensor product to facilitate direct interaction between an input and a hidden state vector from the previous time step. We apply this approach to language modeling, and show how it can improve over long short-term memory (LSTM) models.

2 RECURRENT AND RECURSIVE NEURAL NETWORKS

Given a sequence of input vectors x_t, each of dimension n, an RNN model Elman (1990) consists of a hidden state vector h_t of dimension k, weight matrix W_{hx} with dimensions $k \times n$, a weight matrix W_{hh} with dimensions $k \times k$, and a bias vector b_h of dimension k. The sequence of hidden state vectors is updated according to the equation

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h),$$

where σ is a pointwise nonlinearity such as $1/(1 + \exp(-x))$ or $\tanh$. The model may produce a sequence of output vectors y_t, each of dimension m. In this case, the model also consists of a matrix W_{yh} with dimension $m \times k$ and a bias vector b_y of dimension m. The model produces output vectors according to the equation

$$y_t = W_{yh}h_t + b_y.$$

Recurrent neural networks are typically trained by "unrolling" the model in time and applying the backpropagation algorithm Rumelhart et al. (1986) Werbos (1988).

*Also on the faculty at the University of Nevada, Reno.

A popular variant of RNNs is the long short-term memory (LSTM) network, which mitigates some of the difficulty of training a recurrent model by using "gates" that control the flow of gradient information Hochreiter & Schmidhuber (1997).

2.1 Recursive Neural Tensor Networks

A recursive neural network is structured as a tree: each node in the tree is a transformation that accepts a set of vectors as input and produces a single vector as its output. In a standard recursive neural network the inputs may be combined by concatenation before applying a linear transformation to the result.

Generalizing the simple recursive neural network, the recursive neural *tensor* network uses bilinear tensor products to combine input vectors into a single output vector Socher et al. (2013b) Socher et al. (2013a). Given two input vectors $e_1, e_2 \in \mathbb{R}^d$, the model combines its inputs by computing

$$f(e_1, e_2) = \tanh\left(e_1^T W^{[1:k]} e_2 + V \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b\right),$$

where $W^{[1:k]} \in \mathbb{R}^{k \times d \times d}$ is a tensor and the product $e_1^T W^{[1:k]} e_2$ produces a vector $h \in \mathbb{R}^k$ whose ith component is computed by the equation

$$h_i = e_1^T W^{[i]} e_2,$$

where $W^{[i]}$ is the ith slice of the tensor.

3 Recurrent Tensor Networks

One of the challenges associated with standard recurrent neural network architectures is that at any time t, the input x_t and the history vector h_{t-1} only interact indirectly, through linear combination. In the case of binary tree-structured models, the recursive neural tensor network addresses this problem by using a bilinear tensor product to directly capture the interaction between each node's inputs: the bilinear product allows the two inputs to interact directly, and each slice of the tensor is capable of capturing a different aspect of the inputs' interaction.

We propose the recurrent tensor network (RTN), which updates its hidden state vector using a bilinear tensor product instead of matrix multiplications. In particular, suppose that $h_t \in \mathbb{R}^k$ and $x_t \in \mathbb{R}^n$. Then the vector h_t is updated via the equation

$$h_t = \sigma(h_{t-1} W^{[1:k]} x_t + b_h),$$

where $W^{[1:k]} \in \mathbb{R}^{k \times k \times n}$ is a tensor and the bilinear tensor product $h_{t-1} W^{[1:k]} x_t$ is computed as in Section 2.1. Once we have computed h_t, we can compute an output vector as we did for the models above, using the update equation $y_t = W_{yh} h_t + b_y$.

Gradient-based methods can be used to train a recurrent tensor network. Suppose that L is a loss function defined over the model parameters, and suppose that $q, h \in \mathbb{R}^k$, $x \in \mathbb{R}^n$, $W^{[1:k]} \in \mathbb{R}^{k \times k \times n}$ and $q = h^T W^{[1:k]} x$. Then we find the following derivatives for the ith component of the vector q:

$$\frac{\partial q_i}{\partial x} = (W^{[i]})^T h, \qquad \frac{\partial q_i}{\partial W^{[i]}} = h x^T, \qquad \frac{\partial q_i}{\partial h} = W^{[i]} x.$$

Assuming that we know $\frac{\partial L}{\partial q}$, these derivatives make it straightforward to implement backpropagation through time.

3.1 Gated Recurrent Tensor Networks

The strategy of replacing matrix-vector multiplications with bilinear tensor products is easily extended to recurrent networks that use gates to control gradient flow. For instance, we can produce a tensor-based variant of LSTM using the following update equations:

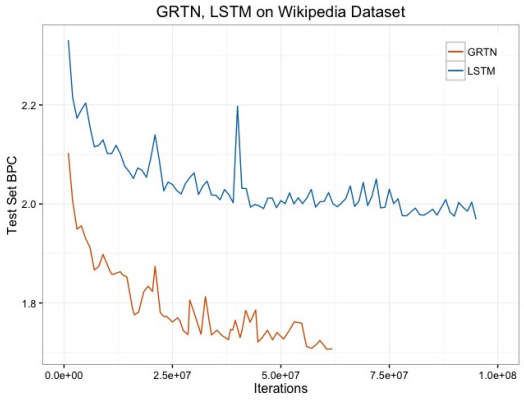

Figure 1: Test set BPC on the Wikipedia data set for both a standard LSTM network and a GRTN.

Model	Test Set BPC
LSTM (3.3 epochs)	1.9852
RTN (3.3 epochs)	1.8720
GRTN (0.6 epochs)	1.7073

Table 1: Test set BPC for LSTM, recurrent tensor networks, and gated recurrent tensor networks. LSTM and RTN ran to convergence. All networks are small, having only 64 hidden units.

$$
\begin{aligned}
i_t &= \sigma(h_{t-1}^T W_i^{[1:k]} x_t + b_i) \\
f_t &= \sigma(h_{t-1}^T W_f^{[1:k]} x_t + b_f) \\
o_t &= \sigma(h_{t-1}^T W_o^{[1:k]} x_t + b_o) \\
\tilde{c}_t &= \tanh(h_{t-1}^T W_{\tilde{c}}^{[1:k]} x_t + b_{\tilde{c}}) \\
c_t &= f_t \circ h_{t-1} + i_t \circ \tilde{c} \\
h_t &= o_t \circ \tanh(c_t),
\end{aligned}
$$

where $W_i^{[1:k]}$, $W_f^{[1:k]}$, $W_o^{[1:k]}$, and $W_{\tilde{c}}^{[1:k]}$ are distinct tensors. We refer to this as a gated recurrent tensor network (GRTN).

4 Evaluation

To evaluate the performance of recurrent tensor networks, we compare their performance against LSTM on the Hutter prize dataset consisting of the first 100M bytes of the English Wikipedia Hutter (2012). We follow the approaches described in Chung et al. (2015) and Graves (2013): we split the 100M byte file into a training set consisting of the first 96M bytes and a test set consisting of the remainder of the file. Each network processes 100 bytes at a time during training. We compare each network using the average number of bits-per-character (BPC) on the test set: $E[-\log_2 P(x_{t+1}|h_t)]$.

We train LSTM, RTN, and GRTN networks, each with a single layer and 64 hidden units. Training was performed with adagrad using an initial learning rate of 0.1 for all models. The LSTM and RTN networks were run to convergence; the GRTN network is significantly slower, and was run on the first 60M bytes of the data.

From Figure 1 we see that, in spite of its runtime performance, the GRTN outperforms an LSTM network. Although this may be attributable to the larger number of parameters in the GRTN, Table 1 shows that the GRTN significantly outperforms the comparably-sized RTN without gates, obtaining a test set BPC of 1.7073 versus the RTN's 1.8720.

5 Conclusion

We have shown that using a bilinear tensor product to combine input and hidden state variables in recurrent networks can lead to a significant improvement in performance for language modeling tasks. Future work will focus on optimizing the runtime performance of this approach, and exploring its viability in other application domains.

REFERENCES

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated Feedback Recurrent Neural Networks. *arXiv:1502.02367 [cs, stat]*, February 2015. URL http://arxiv.org/abs/1502.02367. arXiv: 1502.02367.

Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, March 1990. ISSN 1551-6709. doi: 10.1207/s15516709cog1402_1. URL http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog1402_1/abstract.

Alex Graves. Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*, August 2013. URL http://arxiv.org/abs/1308.0850. arXiv: 1308.0850.

Sepp Hochreiter and Jrgen Schmidhuber. *Long Short-term Memory*. 1997.

Marcus Hutter. €50000 Prize for Compressing Human Knowledge, 2012. URL http://prize.hutter1.net/.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. pp. 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 926–934. Curran Associates, Inc., 2013a. URL http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. 2013b.

Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, January 1988. ISSN 0893-6080. doi: 10.1016/0893-6080(88)90007-X. URL http://www.sciencedirect.com/science/article/pii/089360808890007X.